List Decoding of Algebraic Codes

Nielsen, Johan Sebastian Rosenkilde

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

List Decoding of Algebraic Codes

Johan S. R. Nielsen

Department of
Applied Mathematics and Computer Science

Technical University of Denmark

Abstract

We investigate three paradigms for polynomial-time decoding of Reed–Solomon codes beyond half the minimum distance: the Guruswami–Sudan algorithm, Power decoding and the Wu algorithm. The main results concern shaping the computational core of all three methods to a problem solvable by module minimisation; by applying the fastest known algorithms for this general problem, we then obtain realisations of each paradigm which are as fast or faster than all previously known methods. An element of this is the "2D key equation", a heavily generalised form of the classical key equation, and we show how to solve such using module minimisation, or using our new Demand–Driven algorithm which is also based on module minimisation.

The decoding paradigms are all derived and analysed in a self-contained manner, often in new ways or examined in greater depth than previously. Among a number of new results, we give: a fast maximum-likelihood list decoder based on the Guruswami–Sudan algorithm; a new variant of Power decoding, Power Gao, along with some new insights into Power decoding; and a new, module based method for performing rational interpolation for the Wu algorithm. We also show how to decode Hermitian codes using Guruswami–Sudan or Power decoding faster than previously known, and we show how to Wu list decode binary Goppa codes.

Resumé

Afhandlingen indholder tre paradigmer for afkodning i polynomiel tid af Reed–Solomon-koder ud over den halve minimumsafstand: Guruswami–Sudan, Power, og Wu afkodning. Hovedresultatet er at den beregningsmæssige kerne af alle tre metoder kan udformes som et problem der kan løses med modul-minimering. Ved at benytte de hurtigste kendte metoder til dette generelle problem, opnår vi realiseringer af hvert afkodningsparadigme, som er lige så hurtige eller hurtigere end alle tidligere kendte. Et vigtigt element i udformningen er en kraftigt generaliseret form af den klassiske nøgleligning, de såkaldte "2D nøgleligninger". Vi viser hvordan man kan løse denne slags ligninger ved hjælp af modul-minimering eller ved hjælp af vores nye Demand–Driven-algoritme, som også er baseret på modul-minimering.

I afhandlingen er afkodningsparadigmerne selvstændigt udledt og analyseret, ofte på nye måder eller mere dybdegående end tidligere. Vi præsenterer et antal nye resultater, blandt andet: en hurtig maximum-likelihood listeafkoder baseret på Guruswami–Sudan algoritmen; en ny variant af Power-afkodning, Power Gao, samt nye indsigter i Power afkodning; og en ny, modul-baseret metode for at udføre rational interpolation i Wu algoritmen. Vi viser også hvordan man kan afkode Hermite-koder ved hjælp af Guruswami–Sudan algoritmen eller Power-afkodning som er hurtigere end tidligere kendt, og hvordan man kan Wu listeafkode binære Goppa-koder.

Preface

This work was done under the supervision of Associate Professor Peter Beelen and Professor Tom Høholdt at the Technical University of Denmark in the period July 2010–July 2013. Apart from the present thesis, during this time I authored or coauthored the following four papers:

[BHNW13] P. Beelen, T. Høholdt, J. S. R. Nielsen, and Y. Wu. On Rational-Interpolation Based List-Decoding and List-Decoding Binary Goppa Codes. *IEEE Transactions of Information Theory*, 59(6), 2013.

[NZ13] J. S. R. Nielsen and A. Zeh. Multi-Trial Guruswami–Sudan Decoding for Generalised Reed–Solomon Codes. *Workshop on Coding and Cryptography*, 2013.

[Nie13b] J. S. R. Nielsen. Generalised Multi-sequence Shift-Register Synthesis using Module Minimisation. *IEEE International Symposium on Information Theory*, 2013.

[LSN13] W. Li, V. Sidorenko, and J. S. R. Nielsen. On Decoding Interleaved Chinese Remainder Codes. *IEEE International Symposium on Information Theory*, 2013.

The results of the first three are all contained in this thesis in a similar or generalised form. I did not have time to add the contents of the last article.

Using the open source computer algebra system Sage [S$^+$], I have implemented and conducted experiments with almost all algorithms described in this thesis. They are proof-of-concept implementations, built for ease of investigation and modification rather than speed, and I have striven to make them clear, commented and correct. They are bundled into an open source library called Codinglib [Nie13a], distributed under the GPL v3 licence, and is available for download via `http://jsrn.dk`.

Acknowledgements

I would like to sincerely thank my two advisors Peter Beelen and Tom Høholdt. Peter has given excellent guidance throughout the project; his stash of observations and ideas have often served as inspiration for me delving deeper, but at the same time, he

has fully supported me on my own strange ventures. He has always patiently taken time to assist me when I have barged into his office, stuck in some derivation or needing feedback on a new idea. Tom originally inspired me to pursue mathematics and coding theory; had it not been for his great teaching, I would not have applied for the PhD. Both Peter and Tom believe that a PhD study should be an enjoyable experience, and they have from the beginning been friends as well as advisors. By completely trusting their judgements, I have been able to ward off the depression and most of the stress that unfortunately plague many PhD students. I would also like to thank them both for their encouragement for me to stay in academia, as well as their help in trying to make this possible.

During the Fall 2012 I visited the research group of Daniel Augot at École Polytechnique in Paris. This was a great stay, professionally as well personally. I want to thank Daniel for being a kind and thoughtful host; he made sure I was feeling well at the university, and he assisted me through the French bureaucracy. He even found funding for me to go to YACC 2012 as well as my visit to Ulm University. I hope and trust that we will continue to work together in the future. I also want to thank the rest of the GRACE group, for making me feel very welcome and for helping me learn a bit of French.

I had a one-week visit to Ulm University in November 2012, and I also want to thank the group led by Martin Bossert for welcoming me there. The interest that was shown in my research was immensely encouraging, not to mention fruitful. In particular, thanks to my coauthors Alexander Zeh, Vladimir Sidorenko and Wenhui Li; I enjoyed greatly the research processes for the results of those articles. A big thanks to Martin for offering me a post doc position; I am looking much forward to returning to Ulm.

The pleasant atmosphere and nice colleagues at DTU Mathematics, now DTU Compute, have served to make my studies more enjoyable. The lunch room has seen innumerable discussions on a great variety of subjects, ranging from complex analysis to punctuation rules in Danish. For this, I would like to thank all of the researchers of Building 303B, as well as those who have since moved out or left DTU. In particular, I would like to thank my office mate Fernando Piñero for always having a good excuse for us to take a break; I have enjoyed our many discussions, some of them on math, and our travels together.

I would like to thank my family for their support, encouragement and understanding. To Carola, my love, I am eternally grateful; without you, I would not have dared start on this adventure, and you have kept me confident, grounded, and above all, happy throughout.

12th of July 2013
(Revised 28th of November 2013)

Johan Sebastian Rosenkilde Nielsen

For supporting many of my travels, and in particular several to China, I gratefully acknowledge the support from the Danish National Research Foundation and the National Science Foundation of China (Grant No.11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography.

For supporting my research stay in Paris, I gratefully acknowledge the support from The Otto Mønsted's Fund as well as The Idella Foundation.

Contents

List of floats

Tables

Figures

Algorithms

Chapter 1

Introduction

Polynomial-time list decoding is arguably the greatest advancement of algebraic coding theory in the last two decades: Sudan presented in [Sud97] a conceptually simple, polynomial-time algorithm which permits one to correct errors beyond half the minimum distance for low-rate Generalised Reed–Solomon (GRS) codes. The method finds *all* codewords within a certain radius beyond half the minimum distance, so it realises the conceptual decoding style suggested much earlier by both Elias [Eli57] and Wozencraft [Woz58]. Soon thereafter, Sudan's result was polished into the famous Guruswami–Sudan algorithm in [GS99], allowing list decoding GRS codes for any rate. The algorithm is flexible and simple enough to have been generalised for several classes of codes, e.g. Algebraic Geometric (AG) codes [SW99, GS99], and Chinese Remainder codes [GSS00].

However, "polynomial-time" does not necessarily mean fast enough for practical use, and a tremendous amount of attention by the community has been devoted to finding a faster way to perform the algorithm. It consists of two computationally heavy steps: the "interpolation step", and the "root-finding step". Various methods have been proposed in the last decade for performing these steps in quadratic and even quasi-linear complexity in the code length, e.g. [RR00, LO08, Ale05, BB10].

Simultaneously, two other approaches for decoding beyond half the minimum distance were developed. Schmidt and Sidorenko demonstrated in [SS06] the surprising fact that one can squeeze out *several*—and not just one—of the classical "key equations" involving the error locator when decoding GRS codes of low rate. Solving the key equations simultaneously, one can decode beyond half the minimum distance *most of the time*: there is a slight risk that the key equations "degenerate" and have

solutions of lower degree than the error locator. In practice, this seems to happen so rarely that the eventuality can largely be ignored. The algorithm is thus not a "list decoder", but rather a unique decoder correcting beyond half the minimum distance. The classical single key equation can e.g. be solved using the Berlekamp–Massey algorithm, and building upon the much earlier work by Feng and Tzeng [FT91], Schmidt and Sidorenko proposed a generalisation of this algorithm for solving the list of key equations of their method [SSB10]. The resulting decoder turns out to be able to decode almost exactly as many errors as the original Sudan algorithm. Due to its original derivation, it was termed "decoding by virtual extension to an Interleaved Reed–Solomon code"; in this thesis I have used the name "Power decoding", which seems to already have become a standard label in informal settings.

Wu suggested in [Wu08] a completely different way to continue from the key equation: he noted that even when the number of errors exceed half the minimum distance, then the Berlekamp–Massey algorithm on the single, classical key equation still reveals crucial information: a "small" $\mathbb{F}[x]$-linear combination of the two polynomials calculated from the Berlekamp–Massey algorithm must equal the error locator. By exploiting that the error locator is defined as having zeroes on all the error positions, a fast way to find this linear combination turns out to be a simple generalisation of the core of the Guruswami–Sudan algorithm. Surprisingly, the resulting list decoder can correct exactly the same amount of errors as the Guruswami–Sudan algorithm.

The focus of this thesis is a deeper study of those three paradigms for decoding beyond half the minimum distance. We are in particular concerned with fast methods for executing them on GRS codes. The central approach we have taken for this is to formulate the computationally heaviest part within each paradigm as instances of similar problems, namely that of finding small-degree $\mathbb{F}[x]$ vectors satisfying certain types of linear properties; for Guruswami–Sudan and Wu the computationally heavy part is the interpolation step, and for Power decoding it is the simultaneous solving of the key equations. We will solve the general problem by working with free $\mathbb{F}[x]$-modules, and the strategy can be described in a variety of terms, including Gröbner bases, lattice basis reduction, and module minimisation; all these concepts largely coincide on exactly this kind of algebraic structure. The strategy can be executed by a range of almost standard algorithms from computational algebra: basically we can see such an algorithm as taking any square, full rank matrix over $\mathbb{F}[x]$ and bringing it to a certain "reduced" form called the weak Popov form; this form turns out to directly reveal the solution to our problem. By simultaneously drawing on the description as a Gröbner basis as well as the more linear algebraic one of module minimisation, we immediately have a wealth of intuition and results on the problem and its solutions. By first sending the input matrix through a simple mapping, we can furthermore support a variety of "weighted weak Popov forms"; a flexibility which will be necessary for solving all the types of decoding problems we will meet.

We review three of the fastest algorithms for computing weak Popov forms: Mulders–Storjohann's, Alekhnovich's, and Giorgi–Jeannerod–Villard's (GJV). In particular we show how the first two can be analysed in more detail than previously done to

reveal better performance when a certain measure—the *orthogonality defect*—is low for the input matrix; basically, the number of iterations these algorithms run can be upper bounded using the orthogonality defect. It also turns out that the same two algorithms can be shown to perform well on matrices which are images of the weight-mapping mentioned above, which means that the added flexibility of having weights comes basically for free.

As a natural stepping stone between some of the decoding domain problems and the module minimisation, we propose a family of "2D key equations": a heavy generalisation of the form of the classical key equation. It turns out that the module minimisation problem arising when solving exactly these equations have very low orthogonality defect. Furthermore, by refining the Mulders–Storjohann algorithm, we develop a computationally "demand driven" way to solve such 2D key equations, much akin to the Berlekamp–Massey algorithm, but while retaining the rich underlying algebraic structure of Gröbner bases.

This unified approach has several advantages. In practical terms, it shows how one can support a great many decoding algorithms by having essentially one computational core: module minimisation. One is free to choose any of the three off-the-shelf algorithms mentioned above, and with an optimised implementation of this, one can easily compare the practical performance of all these decoders on concrete code parameters of interest. Such abstraction is much less interesting if it entails computational sub-optimality: but we show how our approach matches or beats essentially *all* the known decoding algorithms for GRS codes. Actually, we point out repeatedly how known methods are special cases of our approach and even step-for-step computationally equivalent.

This leads us to some of the theoretical advantages: with a unified approach, backed by Gröbner bases, we can more immediately and more clearly see properties, connections and generalisations. For instance, several earlier methods have not yet had Divide & Conquer analogues formulated, which is a common, though often laboriously complicated, optimisation technique. Throughout the thesis, we will discuss how several known techniques for a variety of $\mathbb{F}[x]$ linear problems proposed in the coding theory field turn out to essentially be special cases of Mulders–Storjohann; since the Alekhnovich algorithm is exactly a D&C optimisation of Mulders–Storjohann, we are therefore transitively immediately handed D&C optimisations of these earlier techniques. The use of orthogonality defect for counting iterations in complexity estimates is another example: this measure is essentially what is used in several earlier methods, but in a much more ad-hoc fashion; and since we have analysed once and for all the computational complexity of the minimisation algorithms with regards to this, one easily obtains a tighter complexity in all the use cases. In some instances, our general analysis even beats specialised ones seen in the literature.

Yet another advantage, both practical and theoretical, is the flexibility of generalisation to other codes by having formulated these computational methodologies in a unified way. We demonstrate this by decoding Hermitian codes: our framework

immediately gives us the fastest known methods for solving the core computational problem, as well as implying properties such as the decoding radius.

It should be mentioned that module minimisation is not a completely new tool for decoding algorithms, and a few of the applications we give are well-known already. However, this thesis more thoroughly deals with the issue, and demonstrates how module minimisation techniques apply much more universally. It also focus on how the module minimisation algorithm is detached from the minimisation problem, and how each of several standard algorithms perform and have advantages or disadvantages in various settings.

Module minimisation is only an underlying main theme, and the thesis is by no means restricted to this. An effort has been made for the chapters to be self-contained, and we will in particular derive the three paradigms from scratch. We will analyse numerous properties pertaining to the methods, such as decoding radius and parameter choices; often more precisely or in greater detail than previously done. We exploit module minimisation to get a fast Guruswami–Sudan based variant of the list-decoding paradigm, namely the multi-trial decoder. In the case of Power decoding, we give a completely new variant of it, the Power Gao decoder. Guruswami–Sudan and Power decoding of Hermitian codes is investigated, and using our framework we arrive at algorithms which beat all previously known methods. We also reveal new algebraic connections between the decoding paradigms, some of which immediately have practical as well as theoretical applications. Furthermore, we Wu list decode binary Goppa codes, again reaching fast asymptotic complexities.

At the beginning of each chapter we give a more detailed description of the contributions of that chapter.

1.1 Reader's guide

The thesis follows a simple and lightly coupled structure. The remainder of this chapter is devoted to some global remarks on asymptotic analysis, notation, and GRS codes and the error model considered.

Chapter 2 introduces the module minimisation strategy and discuss the three considered algorithms for computing a weak Popov form of a matrix. It also introduces the 2D key equation and how we solve such equations using module minimisation or the new Demand–Driven algorithm. The concepts introduced in Section 2.1 as well as Section 2.5 are necessary for understanding much of the later chapters; intimate knowledge of the module minimisation algorithms, however, is not at all necessary.

Each of three following chapters are devoted to one of the decoding paradigms. They are largely decoupled from each other; in cases where reference to results or definitions in between these chapters are used, page references have been added. All three contain a derivation of the decoding paradigm, analysis of its decoding capability as

well as at least one way to execute it fast using module minimisation. Apart from this, they each describe new extensions, variations and other investigations of the respective paradigm.

Chapter 6 rounds off the thesis with a conclusion and a discussion of future work.

The text contains numerous remarks throughout where I have felt that certain issues or correspondences were worth pointing out. Many examples have been added as well; in Chapter 2 mostly to exemplify definitions, and in the later chapters to exemplify decoding radii or algorithms. The algorithms deal extensively with polynomials and matrices over polynomial rings, and it would be much to lengthy to write examples out in full; therefore, the aim has been to give the reader a sense of the sizes of objects, and only the degrees of the polynomials involved have been given. The main text does not contain many references to related work, and most discussions are deferred to a section devoted to this at the end of each chapter.

1.2 A note on asymptotic analysis

A large part of this thesis is dedicated to investigating fast methods for decoding; by "fast" we mean those with good *worst-case* asymptotic behaviour. Asymptotic analysis for arguing about the complexity of algorithms by nature speaks of infinite *families* of problems: we are interested in the overall behaviour of the complexity, when the size of the problem grows towards infinity. It is designed to be simple to apply yet provide a good overview, and reducing the number of free variables for selecting members of the problem family is important for both. In both these respects, dealing with decoding algorithms is problematic, and the reader should be clear on the limitations in the way we are treating asymptotic analysis in this thesis.

GRS codes exists for any code length n so it is a natural parameter in the asymptotic expressions. Customarily, the dimension k is then chosen as nR for some constant $R < 1$. The decoding radius τ is usually also assumed a constant times n. The alphabet over which the code is defined, however, must grow in size as n grows; in reality, this adds a factor $\log n$ to all expressions, since computing in $\mathbb{F}_{q^m}$ (usually) requires $O(m)$ operations over the field $\mathbb{F}_q$. In practice, these are left out of complexity estimates, and the estimates are always taken to describe the number of operations over the field used. Note that this choice of notation equates e.g. all the expressions $O(k), O(n-k), O(\tau), O(n-\tau)$ and $O(n)$. In practice, we might choose k close to n so $n - k \ll n$; however, this can't readily be expressed in big-O notation. In this thesis, we sometimes express complexities involving such compound expressions like $n - k$, in cases where it is completely clear that the dependence is directly on $n - k$; in all such cases we provide "relaxed" expressions where e.g. this is rewritten as n.

For all decoding algorithms, we will have a parameter ℓ, and for Guruwami–Sudan and Wu also a parameter s. There are a number of possibilities, all of which are

employed somewhere in the literature.

1. We could treat s and ℓ as constants. This leaves them out of the big-O estimates. In practice their impact is quite large, however, so this is clearly very rough.
2. We could estimate s and ℓ in terms of n. We will see in Section 3.1.2 that $s, \ell \in O(n^2)$ and this could be applied; however, such large s and ℓ would never be employed in practice and can easily be avoided. Choosing instead $s, \ell \in O(n)$ discounts decoding very close to the upper bound, and is possibly also more realistic. For some uses $s, \ell \in O(1)$ could even be considered technically sound but see the previous point.
3. We could leave both s and ℓ in the expressions. This is problematic exactly because s and ℓ depend on n for high τ. In Power decoding or Guruswami–Sudan with $s = 1$, then $\ell(k-1) < n$, and in general for Guruswami–Sudan or Wu then s and ℓ are related by a slightly more complicated description; great restrictions must be placed on the allowed use of such rewriting rules in big-O notation if we wish to retain the sought intuition (since, e.g. the above relation allows $O(\ell n^2) = O(\ell k n) = O(n^2)$).
4. We could describe e.g. s in terms of ℓ and leave ℓ in the expressions. We will also see in Section 3.1.2 that $s = \gamma\ell$ for some constant $\gamma < 1$ is usually sensible. But again, in practical settings, γ might be very small, and an algorithm with $O(s^3)$ is obviously better than one with $O(\ell^3)$.

We have opted for leaving both s and ℓ in the expressions, and have avoided any rewriting of the terms from relations such as those mentioned. We just remind the reader that the estimates need to be treated and understood in the proper fashion. When comparing complexities of algorithms, we will usually be considering $n > \ell > s$, i.e. that minimising dependence on n is the primary goal. However, by the nature of our results—i.e. that any module minimisation algorithm can be used for solving the core problems which we will meet—we always present a selection of complexities.

For other codes, e.g. binary Goppa codes, there are similar problems: here we cannot choose n and k independently, and n/k goes to zero for $n \to \infty$. We shortly discuss this in Section 5.3 on page 152 when Wu decoding these codes.

For many complexity estimates, we will draw on fast multiplication techniques. We let $\mathrm{P}(n)$ be the asymptotic computation cost of multiplying together two polynomials of degree at most n over the field in question. For binary fields we have $\mathrm{P}(n) \leq 18n \log n + O(n) = O(n \log n)$ by using the Fast Fourier transformation, while for general fields, using Schönhage–Strassen's algorithm gives $\mathrm{P}(n) = n \log n \log\log n$, see e.g. [vzGG03, Corollary 8.19 and Theorem 8.23]. In practice, Schönhage–Strassen on small to medium degree polynomials performs worse than other methods such as Karatsuba ($O(n^{1.585})$, [KO64]) or Toom–Cook ($O(n^{1.465})$, [Coo66]); I have not been able to find good empirical evidence for the cross-over points for finite field computations, however. There is also the recent algorithm by Fürer ($O(n \log n 2^{\log_2^* n})$, [Für09])

which theoretically beats Schönhage–Strassen on enormous degree polynomials.

Similarly, we let $\mathrm{M}(m)$ be the asymptotic cost of multiplying two $m \times m$ matrices over the field. The trivial method is $O(n^3)$; Strassen's method gives $O(n^{2.81})$ [Str69]; while the fastest known method is William's variant of Coppersmith–Winograd with $O(n^{2.37})$ [Wil12]. In practice, the latter is only faster for extremely large matrices, and Strassen's method also needs quite large matrices to beat the trivial. However, we will mostly be multiplying matrices over $\mathbb{F}[x]$, which means that single-entry multiplication is quite expensive; therefore the asymptotic benefits of the more complicated algorithms should be evident much sooner. Again, I am unfortunately not aware of a serious empirical study of the cross-over points.

For overview, we give "relaxed" versions of all complexities reported in this thesis. There we relax $\mathrm{P}(n)$ into $O(n \log n \log\log n)$. Since our main interest will be rather small matrices, we relax $\mathrm{M}(m)$ into $O(m^3)$. Note that we can multiply together two matrices of $\mathbb{F}[x]^{m\times m}$ with entries of degree at most t, in time $\mathrm{M}(m)\mathrm{P}(t)$.

It is an important and often valid objection to fast polynomial multiplication techniques that the constant hidden in the big-O notation is too high for them to be useful in many applications. Throughout the thesis, we therefore present decoding algorithms with and without the use of fast multiplication.

1.3 Notation

Throughout the thesis we will use certain notational conventions and standard operators. Some of these are completely global, and most of them rather common, so we shortly list them here. The remaining will be introduced at their first usage. The reader should also be aware of the list of symbols and notation, Appendix A, which can be consulted whenever a name is used far from its original definition. The operators given here are also listed in Appendix A.

Vectors are always typeset in bold and most often named in lowercase, e.g. $\boldsymbol{u}$. Matrices are named in uppercase: V. A row of V is named by the same letter in lowercase and indexed: $\boldsymbol{v}_i$. We often won't introduce the rows of a matrix explicitly, but let them automatically be named after this convention; especially in Chapter 2. If $\boldsymbol{v}$ is a vector, then v_j are its elements; the cells of matrices naturally end up with double subscripts: $v_{i,j}$. Except if otherwise introduced, we'll index from 1, so if $\boldsymbol{v}$ has length m its elements are $v_1, \ldots, v_m$.

We extensively work with matrices and vectors over a polynomial ring $\mathbb{F}[x]$ for some field $\mathbb{F}$, and we have a set of notation for this.

- The degree of $\boldsymbol{v} \in \mathbb{F}[x]^m$ is $\deg \boldsymbol{v} = \max_i\{\deg v_i\}$.
- The degree of $V \in \mathbb{F}[x]^{m\times n}$ is $\deg V = \sum_i \deg \boldsymbol{v}_i$, i.e. it's "summed row-degree".
- The *max-degree* of this V is $\operatorname{maxdeg} V = \max_{i,j}\{\deg v_{i,j}\}$.

- The *leading position* of a non-zero vector $\boldsymbol{v} \in \mathbb{F}[x]^m$ is $\operatorname{LP}(\boldsymbol{v}) = \max\{j \mid \deg v_j = \deg \boldsymbol{v}\}$. Note that if more than one entry in $\boldsymbol{v}$ has degree $\deg \boldsymbol{v}$, it is the *rightmost* entry which is the leading position. The *leading term* is the polynomial $\operatorname{LT}(\boldsymbol{v}) = v_{\operatorname{LP}(\boldsymbol{v})}$.
- The *leading coefficient* of a polynomial p is denoted $\operatorname{LC}(p)$.

Example 1.1. *If* $V = \begin{pmatrix} 1 & x^2 \\ x & x \end{pmatrix}$, *then* $\deg V = 3$ *and* $\operatorname{maxdeg} V = 2$. *By convention* $\boldsymbol{v}_1 = \begin{pmatrix} x & x^2 \end{pmatrix}$ *and* $\boldsymbol{v}_2 = \begin{pmatrix} x & x \end{pmatrix}$ *so* $\deg \boldsymbol{v}_1 = 2$ *and* $\deg \boldsymbol{v}_2 = 1$. *Furthermore,* $\operatorname{LP}(\boldsymbol{v}_1) = \operatorname{LP}(\boldsymbol{v}_2) = 2$. ♠

We will also work with multivariate polynomial rings. For a monomial $m = \alpha \prod x_i^{\theta_i} \in \mathbb{F}[x_1, \ldots, x_\kappa]$, then the $(w_1, \ldots, w_\kappa)$-*weighted degree* is $\deg_{w_1,\ldots,w_\kappa} m = \sum w_i \theta_i$, where $w_i \in \mathbb{R}$. For a polynomial in this ring, the $(w_1, \ldots, w_\kappa)$-weighted degree is the maximal of its monomials' $(w_1, \ldots, w_\kappa)$-weighted degrees. We also have two short-hands:

- $\deg P \triangleq \deg_{(1,\ldots,1)} P$. This will also be referred to as "the degree of P".
- $\deg_{x_i} P \triangleq \deg_{(0,\ldots,0,1,0,\ldots,0)} P$ where the 1 is on the ith position.

Lastly, the thesis contains numerous examples; the main purpose is to exemplify the dimensions and relative sizes of the involved objects, rather than being fully calculated examples. For vectors and matrices over $\mathbb{F}[x]$, we therefore only state the elements' *degrees*, and for this introduce the operator $\trianglelefteq$: if $p \in \mathbb{F}[x]$ then $p \trianglelefteq n$ for any $n \geq \deg p$, and $\trianglelefteq$ is then extended element-wise to vectors and matrices. We use the special symbol $\perp$ for which only $0 \trianglelefteq \perp$.

Example 1.2. *Let* V *be as before. Then* $V \trianglelefteq \begin{pmatrix} 0 & 2 \\ 1 & 1 \end{pmatrix}$. ♠

1.4 Generalised Reed–Solomon codes and channel model

For two vectors $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{F}^n$, introduce an *evaluation function* $\operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}} : \mathbb{F}[x] \to \mathbb{F}^n$ by

$$\operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f) = \big(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)\big) \tag{1.1}$$

The codes we will be focusing on in this thesis can then be defined as follows:

Definition 1.3. An $[n, k, d]$ *Generalised Reed-Solomon code*, or GRS code, over a finite field $\mathbb{F}$ is the set

$$\mathcal{C} = \big\{\operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f) \mid f \in \mathbb{F}[x] \wedge \deg f < k\big\}$$

for $\boldsymbol{\alpha} \in \mathbb{F}^n$ with all elements distinct, as well as $\boldsymbol{\beta} \in \mathbb{F}^n$ with all elements non-zero (not necessarily distinct). The α_i are called *evaluation points* and the β_i *column*

multipliers. It is easy to show that $d = n - k + 1$ and the code is therefore MDS. See e.g. [Rot06] for a comprehensive introduction to GRS codes.

We will not be performing any information theoretic calculations, so a formal channel model is not actually necessary. However, the underlying assumptions in many discussions is that we are working over a q-ary symmetric channel with some, relatively low, symbol error probability, or at least a channel which closely resembles this. Here, q is the size of the field of the code. In particular, the clear focus is in hard-decision decoding of individual symbol errors on a field level. Only in related work do we discuss other models, such as burst errors (across field elements) and soft-decision decoding.

A common antipode to the q-ary symmetric channel is Hamming's adversarial channel where the channel is allowed to choose the errors in a malicious way, up to a certain error weight. This harsh channel is fine for list decoding but will render the Power decoding paradigm useless: the channel can force the decoder to fail at any error weight beyond half the minimum distance. Over a random-behaving channel, however, it performs almost perfectly as well as Sudan decoding. Power decoding is the subject of Chapter 4.

Chapter 2

Module minimisation

We will in this thesis extensively deal with free $\mathbb{F}[x]$ modules. Consider such a module $\mathcal{V}$ of dimension m. Any basis of $\mathcal{V}$ can be represented as a list of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{F}[x]^m$, or as a matrix $V \in \mathbb{F}[x]^{m \times m}$ whose rows are the $\boldsymbol{v}_i$. Any element in $\mathcal{V}$ will then be some unique $\mathbb{F}[x]$-linear combination of the rows of V. Similarly, any other basis of $\mathcal{V}$ will as a matrix V' be unimodular equivalent to V, i.e. there exists some $U \in \mathbb{F}[x]^{m \times m}$ with $\det U \in \mathbb{F}$ such that $V' = UV$.

The aim of this chapter is to present algorithms for "minimising" bases of such modules, in the sense of computing from one basis, another basis having a certain form; we show that a basis of this form always exists and has many desirable properties which allow to reason about and solve problems related to finding "small" elements in the module. The reduced basis is essentially a Gröbner basis with respect to a certain module monomial ordering.

It turns out that many decoding algorithms can be formulated to have a problem of exactly this type at its computational core. The remainder of the thesis will be presenting a host of these, and show exactly how the relatively simple and abstract algorithms given in this chapter elegantly handle these cases with little or no additional modelling.

In the next section, we will make precise the hinted notions of "reduced basis" and "small elements", as well as show exactly how these types of bases are Gröbner bases of $\mathcal{V}$. Sections 2.2–2.4 each discuss an algorithm for module minimisation. The algorithms' running times are all different and put weight on different measures of complication of the considered problem, which means that for each there are cases where this is the fastest. In particular, we introduce the measure "orthogonality

defect" upon which the complexity of the first two of the considered algorithms can be shown to depend. In Section 2.5, we will introduce the 2D key equation: a class of equations which strongly generalise the classical key equation of coding theory; we will see many examples from this class—exercising all its generality—in the remainder of the thesis. We show how solving such equations can be done by module minimisation, and in particular how the bases we give have very low orthogonality defect. We give a new algorithm for solving these which is a Demand–Driven variant of the Mulders–Storjohann from Section 2.2. In Section 2.6 we summarise the complexities of the various algorithms for the various problems in tables for future reference. Finally, Section 2.7 discusses related work not already explicitly covered.

In the thesis, we will abuse the term "basis" of an $\mathbb{F}[x]$ module $\mathcal{V}$ of dimension m to additionally refer to any matrix from $\mathbb{F}[x]^{m\times m}$ whose rows constitute a basis of $\mathcal{V}$. Note that such a matrix must have full rank. Furthermore, throughout this chapter we will let $\mathcal{V}$ be such a module of dimension m.

Contributions

- A more precise and general description of how module minimisation, in the sense of weak Popov form, is related to Gröbner bases. Among other things, this has made supporting the very general weights of $\Phi_{\nu,\boldsymbol{w}}$ easy, as well as an improved uncovering of the deep algebraic nature of module minimisation. Proposition 2.14 on page 17 and Proposition 2.26 on page 21 are concrete examples of this.
- The description of the complexity of the Mulders–Storjohann and Alekhnovich algorithms using the orthogonality defect.
- The general form of the 2D key equation, which is one possible description that encompass a long list of equations and approximations, see Table 2.1 on page 28, as well as a host of naturally weighted variants of these.
- How to solve any 2D key equation using module minimisation. This also improves the connection between computational algebra and sequence synthesis, and this chapter contains examples in both directions of how known results from one side are news to the other.
- The Demand–Driven speedup of Mulders–Storjohann for solving 2D key equations is completely new, see Section 2.5.2.
- Theorem 2.35 on page 30 describes precisely the set of all solutions to a 2D key equation once a certain matrix in weak Popov form has been obtained. Naturally this is new, and it generalises and describes in a simple manner similar results from on the set of solutions to more classical key equations and approximations.
- Pointing out that the Alekhnovich algorithm is a Divide & Conquer speedup of Mulders–Storjohann. Also that several module minimisation, Gröbner basis, or sequence synthesis algorithms are special cases of Mulders–Storjohann or

the Demand–Driven algorithm, see Section 2.7.

Many of these ideas, as well as a less general variant of the 2D key equation (with $\rho = 1$) and the Demand–Driven variant of Mulders–Storjohann appeared in [Nie13b]. The complexity of the Alekhnovich algorithm from the orthogonality defect also appeared in [NZ13].

2.1 Preliminaries

2.1.1 Reduced forms

For our notion of "degree" of a matrix, see Section 1.3, the most natural concept of "minimal" basis is the following:

Definition 2.1. A basis V of $\mathcal{V}$ is *row reduced* if $\deg V$ is minimal over all bases of $\mathcal{V}$.

Obviously, any $\mathbb{F}[x]$ module must have a row reduced basis, and such a basis turns out to have many useful properties. However, we will mostly be using a slightly less obvious form, which turns out to be stronger and possess even more interesting properties, but can still be found at essentially the same computational cost.

Definition 2.2. A square matrix V is in *weak Popov* form if an only if the leading position of all rows are different.

One reasons this definition is more useful, computationally, is that it is immediately clear whether a given matrix is in weak Popov form or not, while it seems difficult to tell whether it is row reduced.

Example 2.3. *Consider the following two matrices from* $\mathbb{F}_2[x]^{3\times 3}$*:*

$$A^{(1)} = \begin{pmatrix} x^4+x & 1 & x \\ x^2+1 & x^2 & x \\ x^3 & 1 & x \end{pmatrix} \qquad A^{(2)} = \begin{pmatrix} x & x+1 & x^2+x \\ x^2+1 & x^2 & x \\ x^3 & 1 & x \end{pmatrix}$$

We will see later that the $\mathbb{F}[x]$ *row spaces of* $A^{(1)}$ *and* $A^{(2)}$ *are the same.* $A^{(1)}$ *is not in weak Popov form since it has leading positions* $1, 2, 1$ *respectively.* $A^{(2)}$ *is, however, in weak Popov form since its leading positions are* $3, 2, 1$ *respectively (remember that in case of a tie, such as in row 2, it is the* right-most *position with the row's degree that is leading).* ♠

Unlike row reduced bases, it is not immediately clear whether a given module always admits a basis in weak Popov form. We answer this in the affirmative in Section 2.2 by presenting an algorithm for finding such a matrix. The form, however, is not *unique* for a given basis; for this requirement, we would need the stronger notion of *Popov form*, see e.g. [Pop70, MS03]. This will not be necessary for our aims, though.

Definition 2.4. The *orthogonality defect* of V is $\Delta(V) \triangleq \deg V - \deg\det V$.

The concept of orthogonality defect was introduced for $\mathbb{F}[x]$ matrices by Lenstra [Len85] for estimating the running time of his algorithm on module minimisation; we will use it to a similar effect. The following proposition links together the above definitions, and shows that the orthogonality defect is a measure of the "distance" a matrix has from being in weak Popov form.

Proposition 2.5. *If a square matrix V is in weak Popov form then $\Delta(V) = 0$ and V is row reduced.*

Proof. In the alternating sum-expression for $\det V$, the term $\prod_{i=1}^{m} \mathrm{LT}(\boldsymbol{v}_i)$ will occur since the leading positions of $\boldsymbol{v}_i$ are all different. Thus $\deg\det V = \sum_{i=1}^{m} \deg \mathrm{LT}(\boldsymbol{v}_i) = \deg V$ unless leading term cancellation occurs in the determinant expression. However, no other term in the determinant has this degree: regard some (unsigned) term in $\det V$, say $t = \prod_{i=1}^{m} v_{i,\phi(i)}$ for some permutation $\phi \in S_m$. If not $\phi(i) = \mathrm{LP}(\boldsymbol{v}_i)$ for all i, then there must be an i such that $\phi(i) > \mathrm{LP}(\boldsymbol{v}_i)$ since $\sum_j \phi(j)$ is the same for all $\phi \in S_m$. Thus, $\deg v_{i,\phi(i)} < \deg v_{i,\mathrm{LP}(\boldsymbol{v}_i)}$. As none of the other terms in t can have greater degree than their corresponding row's leading term, we get $\deg t < \sum_{i=1}^{m} \deg \mathrm{LT}(\boldsymbol{v}_i)$.

Thus, $\Delta(V) = 0$. However, the above also proves that the orthogonality defect is at least 0 for *any* matrix. Since any other basis of the same module as that spanned by V is unimodular equivalent to V, it has the same determinant, and so if $\Delta(V) = 0$, V must therefore have minimal degree among these matrices. □

Example 2.6. *Continuing the example of Example 2.3, one can find that* $\det A^{(1)} = \det A^{(2)} = x^7+x^6+x^5+x^2$. *Now* $\deg A^{(1)} = 9$ *and* $\deg A^{(2)} = 7$, *whence* $\Delta(A^{(1)}) = 2$ *and* $\Delta(A^{(2)}) = 0$, *where the latter was indeed expected.* ♠

2.1.2 The equivalence to certain Gröbner bases

We could now proceed to prove how certain "minimal" vectors in a given module $\mathcal{V}$ are found directly from the rows of any basis in weak Popov form, akin to [Ale05, Proposition 2.3] or [Len85, Proposition 1.2]. However, this is just a special case of the more general observation that a basis in weak Popov form is also a Gröbner basis of $\mathcal{V}$ with respect to a certain module monomial ordering. We will even go one step further than this and support a whole family of "weighted" module monomial orderings by considering bases whose images are in weak Popov form after a particular injective mapping. We will also remind the reader of those properties of Gröbner bases which will be important for us in the remainder of the thesis. For a wholesome introduction to Gröbner bases of modules, see e.g. [CLO98].

Gröbner bases and monomial orderings for free modules mimic closely the corresponding theory for ideals and most of the intuition can be carried over. A "monomial" of $\mathcal{V}$ has the form $\alpha x^{\delta} \boldsymbol{e}_i$ where $\alpha \in \mathbb{F}$, $\delta \in \mathbb{N}_0$ and $\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ with the

1 on the i'th position; $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$ is called the standard basis of $\mathbb{F}[x]^m$. A module monomial ordering $\leq$ should then obey rules similar to that of an ideal monomial ordering, i.e. it is a total order on $\mathbb{F}[x]^m$, it is a well-ordering, and if $\boldsymbol{a}_1 \leq \boldsymbol{a}_2$ then $x^\delta \boldsymbol{a}_1 \leq x^\delta \boldsymbol{a}_2$ for all $\delta \in \mathbb{N}_0$, where $\boldsymbol{a}_1, \boldsymbol{a}_2$ are monomials. As with ideals, a module monomial ordering can be extended linearly to an ordering on any element in $\mathbb{F}[x]^m$ (where it is no longer total).

Example 2.7. *Consider $\mathbb{F}[x]^3$, and define a module monomial ordering $\leq$ as follows: for any i, j then $x^{\delta_1}\boldsymbol{e}_i \leq x^{\delta_2}\boldsymbol{e}_j$ if $\delta_1 + i < \delta_2 + j$; to handle when $\delta_i + i = \delta_2 + j$, we define $x^2\boldsymbol{e}_1 \leq \boldsymbol{e}_3 \leq x\boldsymbol{e}_2$ and extend by the rules above. Thus, for example $x^5\boldsymbol{e}_1 \geq x^2\boldsymbol{e}_3 \geq x^2\boldsymbol{e}_2 \geq x\boldsymbol{e}_3 \geq x^3\boldsymbol{e}_1$.* ♠

We will extend our earlier notation so that for any module monomial order $\leq$ and any $\boldsymbol{v} \in \mathbb{F}[x]^m$, $\mathrm{LP}_\leq(\boldsymbol{v})$ is the index i such that $v_i\boldsymbol{e}_i \geq v_j\boldsymbol{e}_j$ for $j \neq i$. Naturally, we also define $\mathrm{LT}_\leq(\boldsymbol{v}) = v_{\mathrm{LP}_\leq(\boldsymbol{v})}$.

We remind the reader that any set of vectors $G \subset \mathbb{F}[x]^m$ is a Gröbner basis with respect to ordering $\leq$ if and only if all S-vectors reduce to zero, where for two $\boldsymbol{v}_i, \boldsymbol{v}_j \in G$

$$S(\boldsymbol{v}_i, \boldsymbol{v}_j) = \frac{t}{\mathrm{LT}_\leq(\boldsymbol{v}_i)}\boldsymbol{v}_i - \frac{t}{\mathrm{LT}_\leq(\boldsymbol{v}_j)}\boldsymbol{v}_j, \qquad \text{where}$$
$$t = \begin{cases} \mathrm{lcm}(\mathrm{LT}_\leq(\boldsymbol{v}_i), \mathrm{LT}_\leq(\boldsymbol{v}_j)) & \text{if } \mathrm{LP}_\leq(\boldsymbol{v}_i) = \mathrm{LP}_\leq(\boldsymbol{v}_j) \\ 0 & \text{otherwise} \end{cases}$$

See e.g. [CLO98, p. 215]

Definition 2.8. A *weighing* is a pair $(\nu, \boldsymbol{w}) \in \mathbb{Z}_+ \times \mathbb{N}_0^m$.

Definition 2.9. For any weighing $\nu, \boldsymbol{w}$, the module monomial ordering $\preceq_{\nu,\boldsymbol{w}}$ is given, for $p_1, p_2 \in \mathbb{F}[x]$, by

$$p_1\boldsymbol{e}_i \preceq_{\nu,\boldsymbol{w}} p_2\boldsymbol{e}_j \quad \Longleftrightarrow$$
$$\big(\nu \deg p_1 + w_i < \nu \deg p_2 + w_j \ \vee \ (\nu \deg p_1 + w_i = \nu \deg p_2 + w_j \wedge i \leq j)\big)$$

Thus, $\preceq_{\nu,\boldsymbol{w}}$ is a flexibly weighted variant of the *term-over-position* ordering with positions ordered naturally, i.e. $x^{w_1}\boldsymbol{e}_1 < \ldots < x^{w_m}\boldsymbol{e}_m$. The most important case in this thesis will be with weighings having $\nu = 1$, so the reader might pay special attention to the simplicity of this case in the remainder of the chapter. However, for the full generality, and in particular the application to Hermitian codes in Section 4.4, we will treat general ν in all results.

Example 2.10. *Continue Example 2.7. It would have been easy had ties been broken in the natural order $x^2\boldsymbol{e}_1 \leq x\boldsymbol{e}_2 \leq \boldsymbol{e}_3$, for then the weighing $\big(1, (1, 2, 3)\big)$ would induce a module monomial ordering $\preceq_{1,(1,2,3)}$ which could be used. As it stands, we can instead employ $\nu > 1$ to achieve the tie breaking:*

$$\nu = 3 \qquad \boldsymbol{w} = (3, 8, 10) = (3 + \epsilon_1, 6 + \epsilon_2, 9 + \epsilon_3)$$

This will suffice since if $\delta_i + i < \delta_j + j$ then also $3\delta_i + 3i + \epsilon_i < 3\delta_j + 3j + \epsilon_j$, since ϵ_i, ϵ_j are both less than 3. Furthermore, if $\delta_i + i = \delta_j + j$ then considering whether or not $3\delta_i + 3i + \epsilon_i < 3\delta_j + 3j + \epsilon_j$ with the chosen ϵ will break the ties exactly as prescribed.

We leave it to the reader to prove that also the weighing $(\nu, \boldsymbol{w}) = \big(2, (0, 3, 4)\big)$ would work. ♠

We then introduce a mapping which will "embed" these weights into vectors of the module in question:

Definition 2.11. For any weighing $\nu, \boldsymbol{w}$, the *weight-embedding mapping* $\Phi_{\nu,\boldsymbol{w}} : \mathbb{F}[x]^m \to \mathbb{F}[x]^m$ is given by

$$\big(v_1(x), \ldots, v_m(x)\big) \mapsto \big(x^{w_1} v_1(x^\nu), \ldots, x^{w_m} v_m(x^\nu)\big)$$

We extend $\Phi_{\nu,\boldsymbol{w}}$ element-wise to sets of vectors, and extend $\Phi_{\nu,\boldsymbol{w}}$ row-wise to $m \times m$ matrices such that the ith row of $\Phi_{\nu,\boldsymbol{w}}(V)$ is $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}_i)$. A matrix V such that $\Phi_{\nu,\boldsymbol{w}}(V)$ is in weak Popov form is said to be in *$\Phi_{\nu,\boldsymbol{w}}$-weighted weak Popov form.*

Note that $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ is a free $\mathbb{F}[x^\nu]$-module of dimension m, though with elements in $\mathbb{F}[x]^m$, and that any basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ is by $\Phi^{-1}_{\nu,\boldsymbol{w}}$ sent back to a basis of $\mathcal{V}$. Also note that if $\nu = 1$, then simply $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}) = \boldsymbol{v} \cdot \operatorname{diag}(x^{w_1}, \ldots, x^{w_m})$.

Proposition 2.12. *If V is a basis of $\mathcal{V}$ in $\Phi_{\nu,\boldsymbol{w}}$-weighted weak Popov form for some weighing $\nu, \boldsymbol{w}$, then V is a Gröbner basis with respect to $\preceq_{\nu,\boldsymbol{w}}$.*

Proof. Note first for any $\boldsymbol{v} \in \mathbb{F}[x]^m$, that $\operatorname{LP}(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v})) = \operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v})$. Thus, since $\Phi_{\nu,\boldsymbol{w}}(V)$ is in weak Popov form, $\operatorname{LT}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v}_i) \neq \operatorname{LT}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v}_j)$ for $i \neq j$. This means that all S-vectors between the rows of V are zero, and so they form a Gröbner basis with respect to $\preceq_{\nu,\boldsymbol{w}}$. □

Example 2.13. *Consider the matrices from Example 2.3 and the weighing $(\nu, \boldsymbol{w}) = (2, (0, 3, 4))$ from Example 2.10. Then*

$$\Phi_{\nu,\boldsymbol{w}}(A^{(1)}) = \begin{pmatrix} x^8 + x^2 & x^3 & x^6 \\ x^4 + 1 & x^7 & x^6 \\ x^6 & x^3 & x^6 \end{pmatrix} \quad \Phi_{\nu,\boldsymbol{w}}(A^{(2)}) = \begin{pmatrix} x^2 & x^5 + x^3 & x^8 + x^6 \\ x^4 + 1 & x^7 & x^6 \\ x^6 & x^3 & x^6 \end{pmatrix}$$

Notice that $\Phi_{\nu,\boldsymbol{w}}(A^{(1)})$ is in weak Popov form while $\Phi_{\nu,\boldsymbol{w}}(A^{(2)})$ is not. We have $\det(\Phi_{\nu,\boldsymbol{w}}(A^{(1)})) = \det(\Phi_{\nu,\boldsymbol{w}}(A^{(2)})) = x^{21} + x^{19} + x^{17} + x^{11}$. As we would expect, $\Delta(\Phi_{\nu,\boldsymbol{w}}(A^{(1)})) = 0$, but also $\Delta(\Phi_{\nu,\boldsymbol{w}}(A^{(2)})) = 0$ even though it is not in weak Popov form. ♠

Remark. A benefit of this modelling is that it is now very easy to show that the algorithms presented later in this chapter for computing bases in weak Popov form can be used for computing Gröbner bases with respect to $\preceq_{\nu,\boldsymbol{w}}$: start with any basis V of $\mathcal{V}$, compute $\Phi_{\nu,\boldsymbol{w}}(V)$ and use this as input to one of the algorithms to compute a matrix in weak Popov form. If this matrix turns out to be in $\Phi_{\nu,\boldsymbol{w}}(\mathbb{F}[x]^{m\times m})$

then it is a basis (over $\mathbb{F}[x^\nu]$) of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$, so applying $\Phi^{-1}_{\nu,\boldsymbol{w}}$ on it we get the sought Gröbner basis. We will show in Proposition 2.26 on page 21 that this last condition is always fulfilled.

The downside is that $\Phi_{\nu,\boldsymbol{w}}(V)$ will contain polynomials of higher degree than V, which means that the complexity bounds for the algorithms will be wider than if running them on V directly. In some cases, the complexity analysis can be improved for modules which are images of $\Phi_{\nu,\boldsymbol{w}}$, however, and the penalty be almost removed; this is the case for several of the algorithms in this chapter. It is not clear if other modellings would have lower penalties for their algorithms. ◆

Having a Gröbner basis in hand, the module division algorithm immediately then gives a host of useful properties. The module division algorithm is almost completely equivalent to the multivariate division algorithm, a critical component of Gröbner basis theory for ideals of multivariate polynomials. Given a Gröbner basis of $\mathcal{V}$ then for a given vector $\boldsymbol{u} \in \mathcal{V}$, the module division algorithm finds the $\mathbb{F}[x]$-linear combination of the basis which constructs $\boldsymbol{u}$. Roughly speaking, it does so by reducing $\boldsymbol{u}$ by subtracting an appropriate multiple of $\boldsymbol{v}_i$, where $\boldsymbol{v}_i$ is a vector in the Gröbner basis having $\operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v}_i) = \operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{u})$, and then recursively reduces the resulting vector until this is zero. For details, consult [CLO98]. By the correctness of this algorithm, it is easy to show that:

Proposition 2.14. *If V is a basis of some $\mathcal{V}$ in $\Phi_{\nu,\boldsymbol{w}}$-weighted weak Popov form for some weighing $\nu, \boldsymbol{w}$, then for any $\boldsymbol{u} \in \mathcal{V}$ there exists unique $p_1, \ldots, p_m \in \mathbb{F}[x]$, which can be found by the division algorithm on $\boldsymbol{u}$ by $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with respect to $\operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}$, such that $\boldsymbol{u} = p_1\boldsymbol{v}_1 + \ldots + p_m\boldsymbol{v}_m$ with also*

$$\begin{aligned}
\deg p_i &\leq \deg u_h - \deg v_{i,h_i} + \nu^{-1}(w_h - w_{h_i}) && \text{if } h_i < h \\
\deg p_i &= \deg u_h - \deg v_{i,h_i} && \text{if } h_i = h \\
\deg p_i &< \deg u_h - \deg v_{i,h_i} + \nu^{-1}(w_h - w_{h_i}) && \text{if } h_i > h
\end{aligned}$$

where $h = \operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{u})$ and $h_i = \operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v}_i)$ for each i.

Proof. The division algorithm on $\boldsymbol{u}$ would commence by reducing $\boldsymbol{u}$ to some $\boldsymbol{u}' \preceq_{\nu,\boldsymbol{w}} \boldsymbol{u}$ by $\boldsymbol{u}' = \boldsymbol{u} - p'_i\boldsymbol{v}_i$ for some $p'_i \in \mathbb{F}[x]$, where i is the unique index such that $h_i = h$. Therefore, $\boldsymbol{u}$ and $p'_i\boldsymbol{v}_i$ must order equal according to $\preceq_{\nu,\boldsymbol{w}}$, which means $\nu \deg u_h + w_h = \nu(\deg p'_i + \deg v_{i,h}) + w_h$ and thus $\deg p'_i = \deg u_h - \deg v_{i,h}$.

In all following iterations of the division algorithm the vector to reduce will order less than $\boldsymbol{u}$ according to $\preceq_{\nu,\boldsymbol{w}}$; thus whenever $\boldsymbol{v}_i$ is used, it must be subtracted with a coefficient of degree at less than $\deg p'_i$. The p_i of the proposition will be the sum of all these coefficients, and thus have degree $\deg p'_i$.

For any other indexes i, we proceed similarly, though the first reduction using $\boldsymbol{v}_i$ with $h_i \neq h$ must be on some $\boldsymbol{u}'' \preceq_{\nu,\boldsymbol{w}} \boldsymbol{u}$. Thus if $p''_i\boldsymbol{v}_i$ is deducted from $\boldsymbol{u}''$, for some $p''_i \in \mathbb{F}[x]$, we must have $p''_i\boldsymbol{v}_i \preceq_{\nu,\boldsymbol{w}} \boldsymbol{u}$. Thus $\nu(\deg p''_i + \deg v_{i,h_i}) + w_{h_i} \leq \nu \deg u_h + w_h$ if $h_i < h$, while $\leq$ is replaced by $<$ otherwise. This holds each iteration that $\boldsymbol{v}_i$ is used. □

Corollary 2.15. *In the context of Proposition 2.14, any* $\boldsymbol{u} \in \mathcal{V}$ *has* $\deg u_{h_i} \geq \deg v_{i,h_i}$ *where* i *is chosen such that* $h_i = \mathrm{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{u})$.

Example 2.16. *Consider the vector* $\boldsymbol{u} = (x^5 + x,\ x^2 + x + 1,\ x^3 + x^2 + x)$. *Since* $A^{(2)}$ *from Example 2.3 on page 13 is in weak Popov form, its rows form a Gröbner basis of its row space with respect to* $\preceq_{1,\mathbf{0}}$. *Running the division algorithm on* $\boldsymbol{u}$, *we first find that since* $\mathrm{LP}(\boldsymbol{u}) = 1 = \mathrm{LP}(\boldsymbol{a}_3^{(2)})$, *we should deduct* $x^2\boldsymbol{a}_3^{(2)}$ *from* $\boldsymbol{u}$ *to get remainder* $\grave{\boldsymbol{u}} = (x,\ x + 1,\ x^2 + x) = \boldsymbol{a}_1^{(2)}$; *thus* $\boldsymbol{u} = \boldsymbol{a}_1^{(2)} + x^2\boldsymbol{a}_3^{(2)}$ *and is in the row space of* $A^{(2)}$. *Proposition 2.14 had predicted* $\deg p_1 \leq 3, \deg p_2 \leq 3, \deg p_3 = 2$, *which are all satisfied.*

Consider now the weighing $(\nu, \boldsymbol{w}) = (2, (0, 3, 4))$ *of Example 2.10; then* $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{u}) = (x^{10} + x^2,\ x^7 + x^5 + x^3,\ x^{10} + x^8 + x^6)$. *Since* $\Phi_{\nu,\boldsymbol{w}}(A^{(1)})$ *is in weak Popov form, the rows of* $A^{(1)}$ *form a Gröbner basis of its row space with respect to* $\preceq_{\nu,\boldsymbol{w}}$. *Performing the* $\preceq_{\nu,\boldsymbol{w}}$*-weighted division algorithm on* $\boldsymbol{u}$ *using the rows of* $A^{(1)}$, *or equivalently, performing the unweighted division algorithm on* $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{u})$ *using the rows of* $\Phi_{\nu,\boldsymbol{w}}(A^{(1)})$ *and then dividing the resulting* x*-powers by* ν, *we find that* $\boldsymbol{u} = \boldsymbol{a}_1^{(1)} + (x^2 + x)\boldsymbol{a}_3^{(1)}$. ♠

Remark. The Gröbner bases of Proposition 2.12 are actually unnormalised *minimal* Gröbner bases (see e.g. [Lau03, p. 212] for minimal Gröbner bases of ideals), since they have a minimal number of elements. They are however not *reduced* (see same reference); these latter are canonical for any free module. Similar to the weak Popov form $\leftrightarrow$ Gröbner basis relation, reduced Gröbner bases with respect to $\preceq_{\nu,\boldsymbol{w}}$ are unsorted variants of the aforementioned Popov form [Pop70, MS03]. ♦

For arguing about running time, we can relate the max-degree and orthogonality defect of bases of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ to those of $\mathcal{V}$:

Lemma 2.17. *For any basis* V *and any weighing* $\nu, \boldsymbol{w}$

$$\begin{aligned} \mathrm{maxdeg}\,(\Phi_{\nu,\boldsymbol{w}}(V)) &\leq \nu\ \mathrm{maxdeg}\,(V) + w_{\max} \\ \Delta(\Phi_{\nu,\boldsymbol{w}}(V)) &\leq \nu\Delta(V) + mw_{\max} - \bar{w} \end{aligned}$$

where $w_{\max} = \max\{w_i\}$ *and* $\bar{w} = \sum_{i=1}^{m} w_i$.

Proof. Note first that for any $\boldsymbol{v} \in \mathcal{V}$, $\deg(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v})) \leq \nu \deg \boldsymbol{v} + w_{\max}$, giving the max-degree bound. It also bounds $\deg(\Phi_{\nu,\boldsymbol{w}}(V))$. Note then that $\det(\Phi_{\nu,\boldsymbol{w}}(V)) = x^{\bar{w}} \det(V)_{|x=x^\nu}$, since it is a sum of terms of the form $\prod_{i=1}^{m} x^{w_{\phi(i)}} v_{i,\phi(i)}(x^\nu)$, where ϕ is a permutation of $1, \ldots, m$. □

Example 2.18. *Consider the matrices of examples Example 2.3 on page 13 and Example 2.10 with* $(\nu, \boldsymbol{w}) = (2, (0, 3, 4))$. *Then*

$$\begin{aligned} \mathrm{maxdeg}\,(\Phi_{\nu,\boldsymbol{w}}(A^{(1)})) = 8 \leq 2\mathrm{maxdeg}\,(A^{(1)}) + 4 &= 12 \\ \mathrm{maxdeg}\,(\Phi_{\nu,\boldsymbol{w}}(A^{(2)})) = 8 \leq 2\mathrm{maxdeg}\,(A^{(2)}) + 4 &= 10 \\ \Delta(\Phi_{\nu,\boldsymbol{w}}(A^{(1)})) = 0 \leq 2\Delta(A^{(1)}) + 3 \cdot 4 - 7 &= 6 \\ \Delta(\Phi_{\nu,\boldsymbol{w}}(A^{(2)})) = 0 \leq 2\Delta(A^{(2)}) + 3 \cdot 4 - 7 &= 5 \end{aligned}$$

♠

2.2 Mulders–Storjohann

We will now present an elegant algorithm for computing a basis in weak Popov form of some free module $\mathcal{V} \subset \mathbb{F}[x]^m$ of rank m, given any square basis of $\mathcal{V}$. The algorithm is due to Mulders and Storjohann [MS03], and was at the time of its publication the fastest known. We will in following sections discuss two other algorithms for solving the same problem, both of which are faster in the generic case for some parameters: the Alekhnovich algorithm, which can be seen as a D&C-variant of the Mulders–Storjohann; as well as the one by Giorgi, Jeannerod and Villard (GJV) which is inherently different. Furthermore, in Section 2.5.2, we will specialise Mulders–Storjohann for certain types of initial bases. The reason for this plethora is that we will not (only) be handling generic cases, and that whichever algorithm is fastest depends on the case.

Both Mulders and Storjohann, Alekhnovich and the GJV were originally given for non-square matrices. However, since we will only be applying them on square, full-rank matrices, and since the complexity estimates for the first two in this case can be improved, we will only present them as such.

Definition 2.19. Applying a *row reduction* on a full-rank matrix over $\mathbb{F}[x]$ means to find two different rows $\boldsymbol{v}_i, \boldsymbol{v}_j$, $\deg \boldsymbol{v}_i \leq \deg \boldsymbol{v}_j$ such that $\operatorname{LP}(\boldsymbol{v}_i) = \operatorname{LP}(\boldsymbol{v}_j)$, and then replacing $\boldsymbol{v}_j$ with $\boldsymbol{v}_j - \alpha x^\theta \boldsymbol{v}_i$ where $\alpha \in \mathbb{F}$ and $\theta \in \mathbb{N}_0$ are chosen such that the leading term of the polynomial $\operatorname{LT}(\boldsymbol{v}_j)$ is cancelled.

Definition 2.20. The *value* function $\psi : \mathbb{F}[x]^m \to \mathbb{N}_0$ is $\psi(\boldsymbol{v}) = m \deg \boldsymbol{v} + \operatorname{LP}(\boldsymbol{v})$.

Lemma 2.21. *If we replace $\boldsymbol{v}_j$ with $\boldsymbol{v}'_j$ in a row reduction, then $\psi(\boldsymbol{v}'_j) < \psi(\boldsymbol{v}_j)$.*

Proof. We can't have $\deg \boldsymbol{v}'_j > \deg \boldsymbol{v}_j$ since all terms of both $\boldsymbol{v}_j$ and $\alpha x^\theta \boldsymbol{v}_i$ have degree at most $\deg \boldsymbol{v}_j$. If $\deg \boldsymbol{v}'_j < \deg \boldsymbol{v}_j$ we are done since $\operatorname{LP}(\boldsymbol{v}'_j) < m$, so assume $\deg \boldsymbol{v}'_j = \deg \boldsymbol{v}_j$. Let $h = \operatorname{LP}(\boldsymbol{v}_j) = \operatorname{LP}(\boldsymbol{v}_i)$. By the definition of leading position, all terms in both $\boldsymbol{v}_j$ and $\alpha x^\theta \boldsymbol{v}_i$ to the right of h must have degree less than $\deg \boldsymbol{v}_j$, and so also all terms in $\boldsymbol{v}'_j$ to the right of h satisfies this. The row reduction ensures that $\deg v'_{j,h} < \deg v_{j,h}$, so it must then be the case that $\operatorname{LP}(\boldsymbol{v}'_j) < h$. □

Algorithm 1 Mulders–Storjohann

Input: A basis V of some free $\mathbb{F}[x]$ module $\mathcal{V} \subseteq \mathbb{F}[x]^{m \times m}$.
Output: A basis of $\mathcal{V}$ in weak Popov form.
1 Apply row reductions on the rows of V until no longer possible.
2 **return** V.

Theorem 2.22. *Algorithm 1 is correct. It performs fewer than $m(\Delta(V) + m)$ row reductions and has asymptotic complexity $O(m^2 \Delta(V) \operatorname{maxdeg} V)$.*

Proof. If Algorithm 1 terminates, the output matrix must be a basis of $\mathcal{V}$ since it is reached by a finite number of row-operations on a basis of $\mathcal{V}$. Since we can apply a

row reduction on a matrix if and only if it is not in weak Popov form, the algorithm must bring V to weak Popov form.

Termination follows directly from Lemma 2.21 since the value of a row decreases each time a row reduction is performed. To be more precise, we furthermore see that the maximal number of row reductions performed on V before reaching a matrix U in weak Popov form is at most $\sum_{i=1}^{m} \psi(\boldsymbol{v}_i) - \psi(\boldsymbol{u}_i)$. Expanding this, we get

$$\begin{aligned}\sum_{i=1}^{m} \psi(\boldsymbol{v}_i) - \psi(\boldsymbol{u}_i) &= \sum_{i=1}^{m} \big(m(\deg \boldsymbol{v}_i - \deg \boldsymbol{u}_i) + \operatorname{LP}(\boldsymbol{v}_i) - \operatorname{LP}(\boldsymbol{u}_i)\big) \\ &= m(\deg V - \deg U) + \sum_{i=1}^{m} \operatorname{LP}(\boldsymbol{v}_i) - \binom{m}{2} \\ &< m(\Delta(V) + m)\end{aligned}$$

where we use $\deg U = \deg \det U = \deg \det V$ and that the $\operatorname{LP}(\boldsymbol{u}_i)$ are all different.

For the asymptotic complexity, note that during the algorithm, no polynomial in the matrix will have larger degree than $\operatorname{maxdeg} V$. The estimate is reached simply by remarking that one row reduction consists of m times scaling and adding two such polynomials. □

Example 2.23. *Consider the matrices of Example 2.3 on page 13; since $A^{(1)}$ is not in weak Popov form, we can use it as input to the Mulders–Storjohann algorithm. Initially, there is only one possible row reduction, namely reducing with $x\boldsymbol{a}_3^{(1)}$ on $\boldsymbol{a}_1^{(1)}$. The result is exactly $A^{(2)}$, so since this is in weak Popov form, the algorithm terminates after only one iteration. This also shows that $A^{(1)}$ and $A^{(2)}$ have the same row space as, previously claimed.* ♠

If we are seeking a Gröbner basis with respect to $\preceq_{\nu,\boldsymbol{w}}$ for some weighing $\nu, \boldsymbol{w}$, the following theorem shows that we can simply run Mulders–Storjohann on $\Phi_{\nu,\boldsymbol{w}}(V)$ for some basis V of $\mathcal{V}$; this is trivial for $\nu = 1$, but if not, we have to verify that we do not leave the $\mathbb{F}[x^\nu]$ module. Since $\Phi_{\nu,\boldsymbol{w}}(V)$ contains larger polynomials than V, one could fret that the running time would suffer; fortunately the theorem also shows that this is not the case.

Theorem 2.24. *Let $\nu, \boldsymbol{w}$ be some weighing and U a basis of $\mathcal{V}$. Algorithm 1 on input $V = \Phi_{\nu,\boldsymbol{w}}(U)$ returns a basis (over $\mathbb{F}[x^\nu]$) of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ in weak Popov form. Let $\delta = \nu^{-1}\Delta(\Phi_{\nu,\boldsymbol{w}}(U))$. The algorithm performs at most $m(\delta + m)$ row reductions. It has complexity $O\big(m^2(\delta+m)\nu^{-1}\operatorname{maxdeg}(\Phi_{\nu,\boldsymbol{w}}(U))\big)$.*

Proof. Since the row reductions are performed over $\mathbb{F}[x]$, we first need to argue that we do not leave the $\mathbb{F}[x^\nu]$ module for V to continue to be a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ after each row reduction: however, to begin with $V = \Phi_{\nu,\boldsymbol{w}}(U)$ is a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$, and since any $\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \in \Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ have $\deg \tilde{u}_i \equiv \deg \tilde{v}_i \mod \nu$ for all i, the x^θ scalar in the first row reduction is a power of x^ν; thus, it is indeed an $\mathbb{F}[x^\nu]$ row reduction, and we still have an $\mathbb{F}[x^\nu]$ basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$. By induction, this will be the case throughout.

To bound the number of row reductions, observe that for any $\boldsymbol{v} \in \mathcal{V}$, if we let $h = \operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v})$ we have

$$\begin{aligned}\psi(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v})) &= m(\nu \deg v_h + w_h) + h \\ &\equiv m w_h + h \mod m\nu\end{aligned}$$

That is, on any given interval of size $m\nu$, $\psi(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}))$ can attain at most m of the values, depending on the value of $\operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v})$. Continuing as in the proof of Theorem 2.22, we get the upper bound on the number of row reductions, which can then be simplified using Lemma 2.17.

Each position of some $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}) \in \Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ is a sparse polynomial with only every ν coefficient non-zero. As before, any monomial in the matrix during the run of the algorithm has at most degree $\operatorname{maxdeg}(\Phi_{\nu,\boldsymbol{w}}(U))$. This gives the complexity $O(m\nu^{-1}\operatorname{maxdeg}(\Phi_{\nu,\boldsymbol{w}}(U)))$ for performing one row reduction. □

Example 2.25. *Consider the weighted matrices of Example 2.13 on page 16; since $\Phi_{\nu,\boldsymbol{w}}(A^{(2)})$ is not in weak Popov form, we can use it as input to the Mulders–Storjohann algorithm. Again, we have only one possible row reduction, reducing with $x^2\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{a}_3^{(2)})$ on $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{a}_1^{(2)})$. The result is exactly $\Phi_{\nu,\boldsymbol{w}}(A^{(1)})$ which is in weak Popov form, whence the algorithm terminates. Notice that the exponent in our row reduction was a power of $\nu = 2$, so it was indeed an $\mathbb{F}[x^\nu]$-combination as expected.* ♠

Now we are in a position to prove that *any* algorithm for bringing matrices to weak Popov form can be used to find a Gröbner basis with respect to $\preceq_{\nu,\boldsymbol{w}}$ of $\mathcal{V}$ using $\Phi_{\nu,\boldsymbol{w}}$. Aside from its possible theoretical interest, it lets us to know that algorithms in the following chapters, in particular the GJV of Section 2.4, work correctly together with $\Phi_{\nu,\boldsymbol{w}}$ without having to investigate in detail the computations.

Proposition 2.26. *Let V be a basis of $\mathcal{V}$. For a given weighing $\nu, \boldsymbol{w}$, let $\tilde{\mathcal{V}}$ be the $\mathbb{F}[x]$-module spanned by the rows of $\Phi_{\nu,\boldsymbol{w}}(V)$ (i.e. the $\mathbb{F}[x]$-closure of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$). If some $\tilde{V}$ in weak Popov form is a basis of $\tilde{\mathcal{V}}$, then $\tilde{V} \in \Phi_{\nu,\boldsymbol{w}}(\mathbb{F}[x]^{m\times m})$ and is a Gröbner basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ (over $\mathbb{F}[x^\nu]$).*

Proof. Let $\tilde{V}_\circ$ be the matrix in weak Popov form returned by Algorithm 1 on input $\Phi_{\nu,\boldsymbol{w}}(V)$. By Theorem 2.24, this is a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ (over $\mathbb{F}[x^\nu]$) and hence also a basis of $\tilde{\mathcal{V}}$ (over $\mathbb{F}[x]$). We will prove the statement by showing how $\tilde{V}_\circ$ can be transformed into $\tilde{V}$ by a series of $\mathbb{F}[x^\nu]$ row operations.

The Popov form is a stronger variant of the weak Popov form, which is canonical for a given free $\mathbb{F}[x]$-module [Pop70]. Mulders and Storjohann in [MS03] gave an algorithm for computing a basis in Popov form, given one in weak Popov form, and this algorithm simply performs a series of row operations on the initial basis (called "simple transformations of the second kind" in the article). These row operations are much like our row reductions in the sense that they use one row to cancel the leading term of some position of another row; thus by the same argument as in the proof of Theorem 2.22, these are in fact $\mathbb{F}[x^\nu]$ row operations.

Since $\tilde{V}$ and $\tilde{V}_\circ$ are both bases of $\tilde{\mathcal{V}}$, they must transform to the same Popov form by this algorithm. Therefore, $\tilde{V}_\circ$ can first be transformed by $\mathbb{F}[x^\nu]$ row operations into Popov form, and then transformed into $\tilde{V}$ by retracing the $\mathbb{F}[x^\nu]$ row operations needed for getting $\tilde{V}$ into Popov form. □

2.3 The Alekhnovich algorithm

Mulders–Storjohann's algorithm admits a Divide & Conquer version which is due to Alekhnovich [Ale05]. However, he seemed not to be aware of the work of Mulders and Storjohann, and that his algorithm is indeed a variant of theirs. A formal description of the algorithm as well as all proofs are in [Ale05]—as well as the more general analysis in [Bra10]—so we will here only give an overview of the algorithm and its connection to Mulders–Storjohann, as well as a discussion on its complexity.

Consider a run of Mulders–Storjohann on input V, a basis of $\mathcal{V}$. The Alekhnovich algorithm works by structuring the same row reductions performed by Mulders–Storjohann in a tree-like fashion; more precisely it hinges on the following series of observations, all of which are proved in [Ale05]:

1. Imagine the row reductions bundled such that each bundle reduces $\operatorname{maxdeg} \tilde{V}$ by 1, where $\tilde{V}$ is the result of applying all earlier row reductions to V.
2. To calculate the row reductions in one such bundle on $\tilde{V}$, one needs for each row $\tilde{\boldsymbol{v}}_i$ of $\tilde{V}$ to know only the monomials in $\tilde{\boldsymbol{v}}_i$ having degree $\deg \tilde{\boldsymbol{v}}_i$.
3. Therefore, to calculate a series of t such bundles, one needs to know only monomials of degree greater than $\deg \tilde{\boldsymbol{v}}_i - t$. Call the matrix containing only these a t-projection of $\tilde{V}$.
4. Any series of row reductions can be represented as an invertible matrix $A \in \mathbb{F}[x]^{m \times m}$ where the product $A\tilde{V}$ is then the result of applying those row reductions to $\tilde{V}$.
5. Thus, we can structure the bundles in a binary tree, where to calculate the row reduction matrix for some node, representing say t bundles, given the matrix $\tilde{V}$, one first recursively calculates the left half of the bundles on a $t/2$-projection of $\tilde{V}$ to get a row reduction matrix A_1. Then recursively calculate the right half of the bundles on a $t/2$-projection of $A_1\tilde{V}$ to get A_2, and the total row reduction matrix becomes $A_2 A_1$.

We have exactly the same choice of row reductions as in Mulders–Storjohann, but the computations are now done on matrices where each cell contains only one monomial (since, in the leaves of the tree, we work on 1-projections), speeding up those calculations by a factor $\operatorname{maxdeg}(V)$. Collecting the row reductions is then done using matrix multiplications.

That the Alekhnovich algorithm can bring $\Phi_{\nu,\boldsymbol{w}}(V)$ to weak Popov form follows immediately from its general correctness, and thus from Proposition 2.26 it produces a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ over $\mathbb{F}[x^\nu]$. However, just as for the case of Mulders–Storjohann,

to better estimate its running time we need to more carefully consider the effects of weights. This is not done in [Ale05], but it was done by Brander [Bra10]. Along with observations similar to those in Section 2.2, for our case we get:

Theorem 2.27. *Let $\nu, \boldsymbol{w}$ be some weighing and U a basis of $\mathcal{V}$. The Alekhnovich algorithm on input $V = \Phi_{\nu,\boldsymbol{w}}(U)$ returns $A \in \mathbb{F}[x^\nu]^{m\times m}$ such that AV is a basis (over $\mathbb{F}[x^\nu]$) of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ in weak Popov form.*

Let $\delta = \nu^{-1}\Delta(\Phi_{\nu,\boldsymbol{w}}(U))$ and assume that $m \in O(\delta)$; then the algorithm has complexity $O\big(\mathrm{M}(m)\mathrm{P}(\delta)\log(\delta)\big)$.

Proof. By [Bra10, Proposition 3.13, Theorem 3.14], computing A has complexity $O(m^2(t+m))$ for performing row reductions, added with $O(m^3\mathrm{P}(\nu^{-1}t)\log(t))$ for matrix multiplications during the algorithm, where t is $\deg(\Phi_{\nu,\boldsymbol{w}}(U)) - \deg(\Phi_{\nu,\boldsymbol{w}}(\tilde{U}))$; for the result to be in weak Popov form, we set $t = \Delta(\Phi_{\nu,\boldsymbol{w}}(U)) = \nu\delta$. However, Brander used in both estimates that the number of row reductions was bounded by $O(m(\nu\delta + m))$; we showed in Theorem 2.24 that it was only $m(\delta + m)$, so going carefully through the proof of Brander, we see that we can actually compute the row reductions in only $O(m^2(\delta + m))$ and the matrix multiplications in $O\big(\mathrm{M}(m)\mathrm{P}(\delta)\log(m(\delta + m))\big)$. □

What we are interested in is not A but AV, however. In the generic case, this product computation turns out to be swallowed by the running time of the Alekhnovich algorithm, but in the case where δ is less than $\nu^{-1}\mathrm{maxdeg}\,(V)$, then the Alekhnovich algorithm will only ever consider a $\nu\delta$-projection of V in order to calculate A. Obviously, to calculate AV, one needs all the terms, but for many cases we actually just need a specific row, and this can give a speedup. The following theorem summarise the complexity in the various cases:

Corollary 2.28. *In the context of Theorem 2.27, and assuming that $m \in O(\delta)$, let $\gamma = \nu^{-1}\mathrm{maxdeg}\,V$. Then the Alekhnovich algorithm can find a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ in weak Popov form in complexity $O\big(\mathrm{M}(m)(\mathrm{P}(\delta)\log(\delta) + \mathrm{P}(\gamma))\big)$.*

Alternatively, assuming only $m \in O(\nu\delta)$, then for any $i \in 1, \ldots, m$, we can find the row with leading position i in complexity $O\big(\mathrm{M}(m)\mathrm{P}(\delta)\log(\delta) + m^2\mathrm{P}(\gamma)\big)$.

Proof. We first compute A in $O\big(\mathrm{M}(m)\mathrm{P}(\delta)\log(\delta)\big)$. In the corollary's first case, we then compute the product AV. By [Bra10, Lemma 3.12], $\mathrm{maxdeg}\,A \leq 2\nu\delta$. Note that both A and V contain only polynomials which are sparse with only every ν coefficient non-zero, and two such polynomials of some degree $\nu\delta$ can be multiplied together in $\mathrm{P}(\delta)$. Therefore, this final matrix product will have complexity at most $O\big(\mathrm{M}(m)\mathrm{P}(\max\{\delta,\ \nu^{-1}\mathrm{maxdeg}\,V\})\big)$.

For the second case, [Ale05, Lemma 2.7] states that the Alekhnovich algorithm would return the same matrix A for V as for $\tilde{V}$, where $\tilde{V}$ is a $\nu\delta$-projection of V. Thus, both $A\tilde{V}$ and AV are in weak Popov form, and the leading position of each row in $A\tilde{V}$ is also the leading position of the corresponding row of AV. Let h be the row in AV with leading position i. We can compute the hth row of AV by first

computing the product $A\tilde{V}$; then finding the value of h by inspecting $A\tilde{V}$; and then computing only the hth row of AV.

Computing $A\tilde{V}$ is performed in $O\big(\mathrm{M}(m)\mathrm{P}(\delta)\big)$ by arguments similar to before, and j is found by simple inspection. Computing the jth row of AV can be done in $O\big(m^2\mathrm{P}(\max\{\delta, \nu^{-1}\mathrm{maxdeg}\, V\})\big)$ by the straightforward algorithm. □

2.4 The GJV algorithm

Giorgi, Jeannerod, and Villard [GJV03] gave a quite different algorithm for bringing matrices to weak Popov form. This algorithm is the fastest known for a generic matrix V, with complexity $O(\mathrm{M}(m)\mathrm{P}(\mathrm{maxdeg}\, V)\log(m\ \mathrm{maxdeg}\, V)^{O(1)})$. It is a Las Vegas certified algorithm, meaning that it is randomised but always returns a correct result in the specified time.

Later in the thesis, we will use this algorithm simply as a black-box solver for finding bases in weak Popov form, and so we will not formally present and prove it; we will here only give a brief introduction to the approach taken by the algorithm.

The main focus in [GJV03] is actually to solve the following generalisation of the Padé approximation problem (see also Section 2.5) and then reduce several polynomial matrix problems—including module minimisation—to that:

Problem 2.29 (Minimal d-approximation basis)**.** Given $V \in \mathbb{F}[x]^{m\times m'}$ and $d \in \mathbb{Z}_+$, find a matrix $M \in \mathbb{F}[x]^{m\times m}$ in weak Popov form such that $MV \equiv 0 \mod x^d$, the right-hand side being the $m \times m'$ zero-matrix.

They give an algorithm with complexity $O\big(\mathrm{M}(m)\mathrm{P}(d)\log(md)^{O(1)}\big)$ for solving this problem: essentially, they show how to build a d'-approximation basis from a $(d'-1)$-approximation basis as well as noting that the identity matrix is a 0-approximation basis. Each iteration operates on $\mathbb{F}[x]^{m\times m}$ matrices of max-degree less than d', and a recurrence relation allows the d iterations to be structured in a Divide & Conquer binary tree and combined using matrix multiplication; this yields the quasi-linear dependence on d.

For reducing module minimisation to the above problem, they do the following: first, one finds $U \in \mathbb{F}[x]^{m\times m}$ such that $H = U^\top(V^\top)^{-1}$ is a proper and irreducible matrix fraction, both of the keywords meaning something similar to what they do for polynomial fractions. Any other proper and irreducible matrix fraction $\tilde{U}^\top(\tilde{V}^\top)^{-1}$ such that $H = \tilde{U}^\top(\tilde{V}^\top)^{-1}$ must have V be unimodular equivalent to $\tilde{V}$. We can also loosen the equality requirement for any d to:

$$\begin{pmatrix}\tilde{V} & \tilde{U}\end{pmatrix}\begin{pmatrix}H^\top \\ -I\end{pmatrix} \equiv 0 \mod x^d \tag{2.1}$$

We can represent H as a matrix in $\mathbb{F}[\![x]\!]^{m\times m}$, i.e. with entries being power series. For any d, we can truncate this representation to x^d in the above equation, making

that equation one purely over $\mathbb{F}[x]$-matrices. The two following observations are generalisations of observations from approximating power series by polynomial fractions: 1) If d is large enough compared to $\operatorname{maxdeg} V$, then the only solutions $\tilde{U}, \tilde{V}$ are those that also satisfy the above with unconditional equality; and 2) a *minimal* solution to this congruence, in a certain precise sense, would imply that $\tilde{V}$ is in weak Popov form.

Now any m rows of a minimal d-appromixation basis of $(H^\top \quad -I)^\top$ will satisfy (2.1). The last piece of the puzzle is then to show that the m minimal-degree rows of such a basis with d chosen big enough will be a minimal solution; and thus from the first n columns of those rows we get a matrix in weak Popov form and unimodular equivalent to V.

In this light, the GJV algorithm resembles a generalised version of a Padé approximation method for polynomials, and in particular the Berlekamp–Massey algorithm. The final result is:

Theorem 2.30. *Given a basis V of $\mathcal{V}$, the GJV Algorithm can compute a basis of $\mathcal{V}$ in weak Popov form in complexity $O\big(\mathrm{M}(m)\mathrm{P}(\operatorname{maxdeg} V)\log(m \operatorname{maxdeg} V)^{O(1)}\big)$.*

Remark. As always, the above complexity is in operations over our ground field $\mathbb{F}$. There is a problem with the Las Vegas initialisation of the algorithm, however, which leads to one needing to perform all computations in a small extension field, as pointed out by Bernstein [Ber11a, p. 4][1]. This adds another factor $\log(\deg\det V) \subset \log(m \operatorname{maxdeg} V)$, which is swallowed by the big-O exponent. ♦

Remark. Since in general, we have $\Delta(V) \in O(m \operatorname{maxdeg} V)$, the GJV algorithm is asymptotically faster than the Alekhnovich by a factor $O(m/\log(m \operatorname{maxdeg} V)^{O(1)})$, which is a good trade-off when m is not very small relative to $\operatorname{maxdeg} V$.

However, for a *random* matrix V, experiments I have conducted with Codinglib [Nie13a] indicate that the row reduction matrices computed in the Alekhnovich algorithm have max-degree orders of magnitude in m from the upper bound used in its complexity estimates, and multiplication of these is what constitutes the most computational expensive part of that algorithm. Furthermore, both the GJV and the Alekhnovich algorithms rely on fast multiplication techniques which notoriously contribute a large constant factor hidden by the big-O notation; however, the GJV relies on it in two levels: to structure the iterations as well as within each iteration.

Coupled with the remark on extension fields, one could therefore imagine that Mulders–Storjohann is fastest on small problems; Alekhnovich fastest on larger or random problems; while the GJV is best on "nasty" and very large problems. However, without a more precise analysis and efficient implementations, this is nothing more than speculation. ♦

[1] Bernstein's note on the GJV algorithm is immediately followed by a complexity comparison between the GJV and the Alekhnovich algorithm; in this comparison Bernstein seem to have misunderstood the complexity of the latter, as he mixes up $\deg V$ with $\operatorname{maxdeg} V$.

An interesting question is then to ask whether we can improve the above complexity estimate of the GJV when the input is of the form $\Phi_{\nu,\boldsymbol{w}}(U)$. This is at least not completely straightforward, and I have not been able to find such an improvement. It is intriguing therefore, that the above algorithm is for parameters of our interest faster than the Alekhnovich for generic matrices and $\nu = 1$, but that whenever $\deg V \in \Omega(\nu\Delta(V))$, this is not necessarily the case. In later chapters we will see examples of both.

2.5 2D key equations

One of the classical decoding methods for Reed–Solomon codes, due to Berlekamp [Ber68], is based on solving the so-called *key equation*:

$$\begin{aligned} \Lambda(x)S(x) &\equiv \Omega(x) \mod x^{n-k} \\ \deg\Lambda &> \deg\Omega \end{aligned} \tag{2.2}$$

where $S(x), n, k$ are known while Λ, Ω are sought such that Λ has minimal degree. We will see much more of this equation later, both in Chapter 4 and Chapter 5, along with a proper description and proof. We will also see a variety of generalisations of it for different decoding methods.

In this section we will therefore define a very broad generalisation of the above and explain how any instance from it can be solved by bringing an appropriate matrix to weak Popov form. Naturally, this job is doable with any of the three algorithms described in preceding sections, but we will also give a new variant of the Mulders–Storjohann algorithm which is faster for most parameter choices but applicable for only this family of modules.

For our generalisation, we have a set of "approximation" criteria such as the above congruence, but the *filtering* on degree requirements will come in two types.

Problem 2.31 (2D key equations, Type 1 and 2)**.** Given

- Size parameters $\rho, \sigma \in \mathbb{Z}_+$
- Moduli $G_1, \ldots, G_\sigma \in \mathbb{F}[x]$, non-zero.
- Polynomials $S_{i,j} \in \mathbb{F}[x]$ with $\deg S_{i,j} < \deg G_j$ for $i = 1, \ldots, \rho$ and $j = 1, \ldots, \sigma$
- Weights $\nu \in \mathbb{Z}_+$, $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_\rho) \in \mathbb{N}_0^\rho$ as well as $\boldsymbol{w} = (w_1, \ldots, w_\sigma) \in \mathbb{N}_0^\sigma$
- If the 2D key equation is of Type 1, then also some $N \in \mathbb{N}$.

Find a non-zero $\boldsymbol{\Lambda} = (\Lambda_1, \ldots, \Lambda_\rho) \in \mathbb{F}[x]^\rho$ such that there exist polynomials $\Omega_1, \ldots, \Omega_\sigma$ satisfying

$$\sum_{i=1}^{\rho} \Lambda_i S_{i,j} \equiv \Omega_j \mod G_j \qquad\qquad j = 1, \ldots, \sigma$$

For Type 1, they should also satisfy

$$\begin{aligned} \nu \deg \Lambda_i + \eta_i &< N, & i &= 1, \dots, \rho \\ \nu \deg \Omega_j + w_j &< N, & j &= 1, \dots, \sigma \end{aligned}$$

For Type 2, they should instead satisfy

$$\max_i \{\nu \deg \Lambda_i + \eta_i\} > \max_j \{\nu \deg \Omega_j + w_j\}$$

If furthermore $\max_i\{\nu \deg \Lambda_i + \eta_i\}$ is minimal among all $\mathbf{\Lambda}$ satisfying the above, for either Type 1 or Type 2, then this is a *minimal* solution.

Example 2.32. *Consider $\mathbb{F}_2[x]$ as our polynomial ring. With $\rho = 2$ and $\sigma = 1$, then*

$$S_1(x) = x^6 + x^5 + x^2 + x \quad S_2(x) = x^6 + x^2 + x \quad G_1(x) = x^7 + x^6 + x^5 + x^2$$

gives rise to a family of 2D key equations of either type and of various weights, where we are searching for $\Lambda_1(x), \Lambda_2(x)$ and $\Omega_1(x)$ such that

$$\Lambda_1(x)S_1(x) + \Lambda_2(x)S_2(x) \equiv \Omega_1(x) \mod G_1(x)$$

We could e.g. seek the one where (Λ_1, Λ_2) has minimal degree (i.e. the greatest of the two polynomials' degrees is minimal) while being greater than $\deg \Omega_1$. That would correspond to Type 2 and the weights $\nu = 1$ and $\eta_1 = \eta_2 = w_1 = 0$. ♠

We will call both the vectors $(\Lambda_1, \dots, \Lambda_\rho)$ and the complete $(\Lambda_1, \dots, \Lambda_\rho, \Omega_1, \dots, \Omega_\sigma)$ *solutions* to the 2D key equation if they satisfy the requirements. Since all the requirements in Problem 2.31 for the indeterminates are $\mathbb{F}$-linear, all solutions form an $\mathbb{F}$-linear space; in a sense, within the degree limitations, it is even $\mathbb{F}[x]$-linear.

Remark. It would be an equally powerful definition to omit the Ω_j and always have the right-hand side 0. Ignoring the filtering condition for now, a 2D key equation could still be described by this, since if we define for $i = \rho + 1, \dots, \rho + \sigma$ that $S_{i,j} = -1$ for $j = i + \rho$ and $S_{i,j} = 0$ otherwise, then

$$\sum_{i=1}^{\rho} \Lambda_i S_{i,j} + \sum_{i=\rho+1}^{\rho+\sigma} \Omega_{i-\rho} S_{i,j} \equiv 0 \mod G_j$$

so $(\Lambda_1, \dots, \Lambda_\rho, \Omega_1, \dots, \Omega_\sigma)$ is a solution to a 2D key equation with $\rho + \sigma$ left-hand side indeterminates and all right-hand side indeterminates forced zero by w_i set sufficiently large.

A problem definition with all the right-hand side always zero would closely resemble a weighted, twisted (by the G_j) variant of minimal d-approximant. While this is more elegant than our definition in some sense, our module interpretation more directly fits our definition (by a factor 2 in module dimension). Furthermore, for the

Name	Specialisations	References
Simultaneous Padé	$\rho = 1$, $G_j = x^{d_j}$, fixed N	[BGM96, BL92, BL09]
Rational reconstruction	$\rho = 1$, $G_j = G$, $w_j = w$	[OS06]
Hermite Padé	$\sigma = 1$, $G_1 = x^d$, fixed N	[BGM96, BL92]
Matrix Padé[a]	$G_j = x^d$, fixed N	[BL92, BL94, BGM96]
Extended key equation	$\sigma = 1$, $G_1 = x^d$	[RR00]
Generalised key equations	$G_j = x^{d_j}$	[ZGA11]
Classical key equation	$\rho = \sigma = 1$, $G_1 = x^d$, $\eta_1 = w_1 = 0$	[Ber68, Mas69]
Goppa key equation	$\rho = \sigma = 1, \eta_1 = w_1 = 0$	[SKHN75]
Gao key equation	$\rho = \sigma = 1$	[Gao03]
Power key equations *or* Multi-sequence LFSR	$\rho = 1$, $G_j = x^{d_j}$	[FT91, SS06, Wang[3]]
Minimal d-approximation	$G_j = x^d$, $w_j = d$	[BL92, BL94, GJV03]

Table 2.1: Several types of equations and approximations discussed in the literature which 2D key equations generalise. The first group is of Type 1 and the second of Type 2. Setting an indexed variable to a non-indexed one, e.g. $G_j = G$, means that all of the variables must have the same value. By "fixed N" we mean that η_i, w_j and N are assigned such that a solution is always guaranteed to exist by linear algebraic arguments. In all of the above $\nu = 1$. The references are not necessarily exhaustive lists on articles concerning each type.

[a] In [BL92], this is called "Vector Hermite Padé". For slightly less restricted η_i, it is called "Power Hermite Padé" in [BL92, BL94]

applications in coding theory given in this thesis, we will always have a non-zero right-hand side as ours. ◆

Problem 2.31 is not only a generalisation of the key equation (2.2), but of a range of other types of equations and Padé approximation variants. Table 2.1 gives an overview of some of these as well as how their form is specialised from the 2D key equations, but see also Section 2.7

Remark. As can be seen in Table 2.1, we can find a minimal d-approximation by solving a 2D key equation; in fact, as we will see in Section 2.5.1, we can find an entire minimal d-approximation basis. Not surprisingly, we will be solving such 2D key equations by finding a basis in weak Popov form of a certain module.

We described in Section 2.4 how the GJV algorithm works by finding a minimal d-approximation basis of a certain module, and how this can be used to find bases in weak Popov form. Thus, we could, for instance, solve a minimal d-approximation problem by solving a 2D key equation by performing module minimisation, by finding a minimal d-approximation basis. The modules that are manipulated double in dimension for each indirection in this loop, though, so apart from proving a

certain computational equivalence between module minimisation and minimal d-approximation (under the given transformations), this is mostly a curiosity. ♦

Example 2.33. *It is perhaps not immediately clear that $\nu > 1$ is ever useful, and indeed* all *the uses of 2D key equations in this theses apart from Section 4.4 employ $\nu = 1$. Indeed, for a 2D key equation of Type 1, it is* never *necessary to use $\nu > 1$: with a 2D key equation of Type 1 and $\nu > 1$, it is not too hard to show that one can always find new weights such that choosing these with $\nu = 1$ allows exactly the same set of solutions. It is therefore somewhat superfluous to allow $\nu > 1$ in the definition for this case.*

However, the same does not hold for Type 2: the following demonstrates by a contrived example that it can be useful to find solutions to the congruence satisfying quite specific degree constraints, while disallowing others. Let us focus only on the degrees of the Λ_i and Ω_j, and let $\sigma = \rho = 2$. Assume therefore that $\boldsymbol{s} = (\Lambda_1, \Lambda_2, \Omega_1, \Omega_2)$ satisfies the congruence. Now we wish to recognise it as a solution if also

$$\boldsymbol{s} \trianglelefteq (1,1,1,1) \qquad \text{or} \qquad \boldsymbol{s} \trianglelefteq (0,1,0,1)$$

but we do not *wish to recognise it as a solution if it instead satisfied*

$$\boldsymbol{s} \trianglelefteq (0,0,0,1) \qquad \text{or} \qquad \boldsymbol{s} \trianglelefteq (0,1,1,0)$$

Note that if $\boldsymbol{s} \trianglelefteq (1,1,1,1)$ is recognised as a solution, that implies for instance that also $\boldsymbol{s} \trianglelefteq (2,2,2,2)$ is recognised as a solution, due to the way the weights filters solutions in Type 2. Now, choosing $\nu = 4$, one can easily verify that the above is fulfilled by choosing $(\eta_1, \eta_2, w_1, w_2) = (3,1,2,0)$. We will show that with $\nu = 1$, it is not possible to achieve the same: in this case, since $\boldsymbol{s}$ is a solution if and only if $\max\{\deg \Lambda_i + \eta_i\} > \max\{\deg \Omega_j + w_j\}$, then the restrictions imply that the weights must satisfy

$$\begin{aligned} \max\{w_1, w_2\} < \quad \max\{\eta_1, \eta_2\} \quad &\leq \max\{w_1, w_2 + 1\} \\ \max\{w_1, w_2 + 1\} < \max\{\eta_1, \eta_2 + 1\} &\leq \max\{w_1 + 1, w_2\} \end{aligned}$$

But that means $\max\{w_1, w_2\} < \max\{w_1, w_2 + 1\} < \max\{w_1 + 1, w_2\}$. This is impossible to achieve since w_1, w_2 are integers.

In Section 4.4 we use $\nu > 1$ and actually demonstrate that it is quite necessary for optimal decoding performance, see the remark on page 121. ♠

2.5.1 Module perspective

Consider a given 2D key equation problem. We will now give a free $\mathbb{F}[x]$ module which has the property that any basis in weak Popov form directly reveals a minimal solution to the problem; we even get something like a Gröbner basis for the space of *all* solutions.

Consider $M \in \mathbb{F}[x]^{(\rho+\sigma)\times(\rho+\sigma)}$ such that

$$M = \left(\begin{array}{c|c} I & S \\ \hline 0 & G \end{array}\right) = \left(\begin{array}{cccc|ccc} 1 & 0 & \dots & 0 & S_{1,1} & \dots & S_{1,\sigma} \\ 0 & 1 & & \vdots & S_{2,1} & S_{2,2} & & S_{2,\sigma} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots \\ 0 & \dots & & 1 & S_{\rho,1} & S_{\rho,2} & \dots & S_{\rho,\sigma} \\ \hline 0 & 0 & \dots & 0 & G_1 & 0 & \dots & 0 \\ 0 & 0 & & \vdots & 0 & G_2 & & \vdots \\ \vdots & & \ddots & & \vdots & & \ddots & \\ 0 & \dots & & 0 & 0 & \dots & & G_\sigma \end{array}\right) \tag{2.3}$$

where I is the $\rho \times \rho$ identity matrix, $S \in \mathbb{F}[x]^{\rho\times\sigma}$ has elements $S_{i,j}$, and $G = \operatorname{diag}(G_1, \dots, G_\sigma)$. Let $\mathcal{M}$ be the $\mathbb{F}[x]$ module spanned by the rows of M.

Lemma 2.34. *A vector $(\lambda_1, \dots, \lambda_\rho, \omega_1, \dots, \omega_\sigma)$ is in $\mathcal{M}$ if and only if*

$$\sum_{i=1}^{\rho} \lambda_i S_{i,j} \equiv \omega_j \mod G_j \qquad\qquad j = 1, \dots, \sigma \tag{2.4}$$

Proof. Equation (2.4) is easily seen to be satisfied for each row of M, so by $\mathbb{F}[x]$-linearity, it is satisfied for every vector in $\mathcal{M}$. Contrarily, for any vector $\boldsymbol{v} = (\lambda_1, \dots, \lambda_\rho, \omega_1, \dots, \omega_\sigma)$ satisfying (2.4) there must exist $p_1, \dots, p_\sigma \in \mathbb{F}[x]$ such that $\sum_{i=1}^{\rho} \lambda_j S_{i,j} + p_i G_i = \omega_i$ wherefore $\boldsymbol{v} = \lambda_1 \boldsymbol{m}_1 + \dots + \lambda_\rho \boldsymbol{m}_\rho + p_1 \boldsymbol{m}_{\rho+1} + \dots + p_\sigma \boldsymbol{m}_{\rho+\sigma}$, and so $\boldsymbol{v} \in \mathcal{M}$. □

Denote by $\bar{\boldsymbol{w}}$ the vector of all the weights $(\eta_1, \dots, \eta_\rho, w_1, \dots, w_\sigma)$ and by $\bar{w}_i$ the elements of this vector.

Theorem 2.35. *Let V be a basis of $\mathcal{M}$ in $\Phi_{\nu,\bar{\boldsymbol{w}}}$-weighted weak Popov form with the rows ordered such that $\operatorname{LP}_{\preceq_{\nu,\boldsymbol{w}}}(\boldsymbol{v}_i) = i$. Then for any non-zero $\boldsymbol{u} \in \mathcal{M}$ there exists unique $p_1, \dots, p_{\rho+\sigma} \in \mathbb{F}[x]$, which can be found by the division algorithm on $\boldsymbol{u}$ by $\boldsymbol{v}_1, \dots, \boldsymbol{v}_{\rho+\sigma}$ with respect to $\operatorname{LP}_{\preceq_{\nu,\bar{\boldsymbol{w}}}}$, such that $\boldsymbol{u} = \sum_{i=1}^{\rho+\sigma} p_i \boldsymbol{v}_i$.*

1. *$\boldsymbol{u}$ is a solution to the 2D key equation of Type 1 if and only if for all i then*

$$\deg p_i < \nu^{-1}(N - \bar{w}_i) - \deg(v_{i,i}) \tag{2.5}$$

2. *$\boldsymbol{u}$ is a solution to the 2D key equation of Type 2 if and only if for $i > \rho$ then*

$$\deg p_i < \max_{j \le \rho} \{\deg p_j + \deg v_{j,j} + \nu^{-1}\bar{w}_j\} - \nu^{-1}\bar{w}_i - \deg v_{i,i} \tag{2.6}$$

Proof. Obviously p_i exist such that they linearly combine with the $\boldsymbol{v}_i$ into $\boldsymbol{u}$, since the $\boldsymbol{v}_i$ form a basis of $\mathcal{M}$. For the degree constraints, recall that by Proposition 2.12 on page 16, the $\boldsymbol{v}_i$ form a Gröbner basis under the ordering $\preceq_{\nu,\bar{\boldsymbol{w}}}$. From Lemma 2.34 we know that all rows of V must satisfy (2.4). Note also that for any $\boldsymbol{u}$, then $\nu \deg u_i + \bar{w}_i = \deg(x^{\bar{w}_i} u_i(x^\nu))$ which means the maximal of all of these equals $\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{u}))$.

Consider first Type 1. By the above $\boldsymbol{u}$ is a solution if and only if $\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{u})) < N$. But $\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(p_i\boldsymbol{v}_i)) < N \iff p_i < \nu^{-1}(N - \bar{w}_i) - \deg(v_{i,i})$, so $\boldsymbol{u}$ is a solution if and only if it satisfies the requirement (2.5).

Consider now Type 2 and let $h = \mathrm{LP}_{\preceq_{\nu,\bar{\boldsymbol{w}}}}(\boldsymbol{u})$. By the above, $\boldsymbol{u}$ is a solution if and only if $h \leq \rho$ in which case $\deg u_h = \max_{j\leq\rho}\{\deg p_j + \deg v_{j,j} + \nu^{-1}\bar{w}_j\} - \nu^{-1}\bar{w}_h$. By Proposition 2.14 on page 17, we have for $i > \rho$ that $\deg p_i < \deg u_h - \deg v_{i,i} + \nu^{-1}(\bar{w}_h - \bar{w}_i)$, and the theorem follows. □

The theorem therefore gives, for both 2D key equations of Type 1 or 2, a way to enumerate the space of all solutions; in this sense, the rows of V constitute a "basis" of the solutions, though obviously not in the usual sense of basis since the $\mathbb{F}[x]$-linear combinations are degree-wise restricted. Due to the rows of V being a Gröbner basis of the entire space $\mathcal{M}$, the division algorithm also gives us an easy way to find for any given solution its $\mathbb{F}[x]$-linear construction from the rows of V.

Corollary 2.36. *If V is a basis of $\mathcal{M}$ in $\Phi_{\nu,\bar{\boldsymbol{w}}}$-weighted weak Popov form, then*

1. *the row $\boldsymbol{v}$ of V which is minimal according to $\preceq_{\nu,\bar{\boldsymbol{w}}}$ is a minimal solution to the 2D key equation of Type 1 if any solutions exists.*
2. *the row $\boldsymbol{v}$ of V which is minimal according to $\preceq_{\nu,\bar{\boldsymbol{w}}}$ while $\mathrm{LP}_{\preceq_{\nu,\bar{\boldsymbol{w}}}}(\boldsymbol{v}) \leq \rho$ is a minimal solution to the 2D key equation of Type 2.*

Proof. Since the rows of V constitute a Gröbner basis of $\mathcal{M}$ with respect to $\preceq_{\nu,\bar{\boldsymbol{w}}}$, any element $\boldsymbol{u} = p_1\boldsymbol{v}_1 + \ldots + p_{\rho+\sigma}\boldsymbol{v}_{\rho+\sigma}$ orders equal according to $\preceq_{\nu,\bar{\boldsymbol{w}}}$ as the greatest of the $p_i\boldsymbol{v}_i$, i.e. no leading term cancellation occurs in the sum of the $p_i\boldsymbol{v}_i$. The corollary then follows directly from Theorem 2.35. □

Example 2.37. *Continue Example 2.32 on page 27. The corresponding matrix would be*

$$M = \begin{pmatrix} 1 & 0 & S_1 \\ 0 & 1 & S_2 \\ 0 & 0 & G_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & x^6 + x^5 + x^2 + x \\ 0 & 1 & x^6 + x^2 + x \\ 0 & 0 & x^7 + x^6 + x^5 + x^2 \end{pmatrix}$$

For the considered weights, we are to module minimise $\Phi_{1,\mathbf{0}}(M) = M$. Running any of the algorithms discussed in the preceding sections, we might end up with a variety of different matrices, since the weak Popov form is non-canonical. One of these possible matrices is $A^{(2)}$ of Example 2.3 on page 13. Using Corollary 2.36 we can read off $A^{(2)}$ that $\boldsymbol{a}_2^{(2)} = (x^2 + 1, x^2, x) = (\Lambda_1, \Lambda_2, \Omega_1)$ is a minimal solution to the 2D key equation. The reader can verify that indeed $\Lambda_1 S_1 + \Lambda_2 S_2 \equiv \Omega_1 \mod G_1$. From Theorem 2.35 we can also realise that there are no other minimal solutions: (2.6) disallows a contribution from $\boldsymbol{a}_1^{(2)}$ since it has degree 2, and $\boldsymbol{a}_3^{(2)}$ is immediately out since it has degree 3. This also implies that $\boldsymbol{a}_2^{(2)}$ must be a row in any *weak Popov basis of the row space of $A^{(2)}$.*

Not surprisingly, we can also go from a weighing $\nu, \boldsymbol{w}$ to the weights of a 2D key equation. Having the same polynomials as above and using the weighing $(\nu, \boldsymbol{w}) =$

$(2,(0,3,4))$ *of Example 2.10 on page 15 would correspond to seeking* $\Lambda_1, \Lambda_2, \Omega_1$ *satisfying the congruence as well as* $\max\{2\deg\Lambda_1,\ 2\deg\Lambda_2+3\} > 2\deg\Omega_1+4$. *If this was of interest, one would module minimise* $\Phi_{\nu,\boldsymbol{w}}(M)$ *and could possibly end up with* $\Phi_{\nu,\boldsymbol{w}}(A^{(1)})$, *listed in Example 2.13 on page 16. From here, we read off that*

$$\Phi^{-1}_{\nu,\boldsymbol{w}}(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{a}^{(1)}_2)) = \Phi^{-1}_{\nu,\boldsymbol{w}}\big((x^4+1,\ x^7,\ x^6)\big) = (x^2+1, x^2, x)$$

is a minimal solution to also this 2D key equation. However, this time it is not unique, since adding $\boldsymbol{a}^{(1)}_3$ *gives another minimal solution* $(x^3+x^2+1,\ x^2+1,\ 0)$. ♠

In order to argue about the running time of applying the module minimisation algorithms from preceding sections, we need a bound on the orthogonality defect of M. The generic bound for any square matrix would only give $\Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) \leq m\ \mathrm{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M))$, but the special form of M means its orthogonality defect is much lower: (below, recall that $\mathrm{pos}(x) = x$ for $x > 0$ and $\mathrm{pos}(x) = 0$ for $x \leq 0$)

Lemma 2.38.

$$\begin{aligned}
\mathrm{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) &= \max\{\eta_1,\ldots,\eta_\rho, \nu\deg G_1 + w_1, \ldots, \nu G_\sigma + w_\sigma\}\\
\deg\det(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) &= \sum_{i=1}^{\rho}\eta_i + \sum_{j=1}^{\sigma}\big(\nu\deg G_j + w_j\big)\\
\Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) &= \sum_{i=1}^{\rho}\mathrm{pos}(\max_j\{\nu\deg S_{i,j} + w_j\} - \eta_i)\\
&\leq \rho(\mathrm{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) - 1) - \textstyle\sum_{i=1}^{\rho}\eta_i
\end{aligned}$$

Proof. Due to the structure of M, we have

$$\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) = \sum_{i=1}^{\rho}\max\{\eta_i, \nu\deg S_{i,1}+w_1, \ldots, \nu\deg S_{i,\sigma}+w_\sigma\} + \sum_{j=1}^{\sigma}\big(\nu\deg G_j + w_j\big)$$

Since $\Phi_{\nu,\bar{\boldsymbol{w}}}(M)$ is a diagonal matrix, its determinant is the product of its diagonal entries and so

$$\deg\det(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) = \sum_{i=1}^{\rho}\eta_i + \sum_{j=1}^{\sigma}\big(\nu\deg G_j + w_j\big)$$

The orthogonality defect is the difference between these.

The statement on the max-degree follows directly from the structure on M, and remembering $\deg S_{i,j} < \deg G_j$ for all j. □

Using two quite different techniques, we can upper bound the degree of a minimal solution to a 2D key equation depending on its type. The case of Type 2 turns out to be much more complicated and not even completely determinable; unfortunately, this makes the following few pages highly technical. We introduce a simple operator, which we will use repeatedly throughout the thesis: $\mathrm{pos}(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ where $\mathrm{pos}(x) = x$ if $x > 0$ and $\mathrm{pos}(x) = 0$ for $x \leq 0$. Then:

Proposition 2.39. *Consider a given 2D key equation with a minimal solution $\boldsymbol{s}$. If the considered equation is of Type 1 then $\boldsymbol{s}$ satisfies*

$$\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{s})) \leq \deg\det(\Phi_{\nu,\bar{\boldsymbol{w}}}(M))/(\rho+\sigma)$$

If it is of Type 2 then $\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{s})) \leq d$, where d is the least integer satisfying

$$\sum_{j=1}^{\sigma} \operatorname{pos}(\deg G_j - \lceil \nu^{-1}(d-w_j)\rceil) \leq \sum_{i=1}^{\rho} \lfloor \nu^{-1}(d-\eta_i)\rfloor + (\rho-1),$$

except in the case $\operatorname{rank} U' = \operatorname{rank} U$, where U is a certain matrix over $\mathbb{F}$ pertaining to the 2D Key Equation and d, and U' is U with a certain column removed, both defined in the proof.

Proof. For Type 1, we know that if $\Phi_{\nu,\bar{\boldsymbol{w}}}(V)$ is a basis of $\Phi_{\nu,\bar{\boldsymbol{w}}}(\mathcal{M})$ in weak Popov form, then $\deg \Phi_{\nu,\bar{\boldsymbol{w}}}(V) = \deg\det \Phi_{\nu,\bar{\boldsymbol{w}}}(V)$ from which it follows that the row $\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{s})$ with minimal degree must have $\deg \Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{s}) \leq \frac{1}{\rho+\sigma} \deg\det \Phi_{\nu,\bar{\boldsymbol{w}}}(V)$.

For Type 2, since solutions must satisfy $\operatorname{LP}_{\preceq_{\nu,\bar{\boldsymbol{w}}}}(\boldsymbol{s}) \leq \rho$, we proceed differently and instead go back to the 2D key equation definition. Let $\boldsymbol{v} = (\Lambda_1,\dots,\Lambda_\rho,\Omega_1,\dots,\Omega_\sigma)$ be a vector which is a solution to the 2D key equation and let $d = \deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{v}))$; we are trying to upper bound d so that we are sure such a $\boldsymbol{v}$ exists; then since $\boldsymbol{s} \preceq_{\nu,\bar{\boldsymbol{w}}} \boldsymbol{v}$ we have $\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{s})) \leq d$. Now, $\deg\Lambda_i \leq \nu^{-1}(d-\eta_i)$ and $\deg\Omega_j < \nu^{-1}(d-w_j)$, but remember that at least one of the Λ_i must satisfy the above with equality for $\boldsymbol{v}$ to satisfy the leading position restriction.

For each j, we have $\sum_{i=1}^{\rho} \Lambda_i S_{i,j} \equiv \Omega_j \mod G_j$ which is equivalent to there existing $q_j \in \mathbb{F}[x]$ such that $\sum_{i=1}^{\rho} \Lambda_i S_{i,j} - q_j G_j = \Omega_j$. We can therefore change focus and say instead that we are searching Λ_i and q_j such that $\deg(\sum_{i=1}^{\rho} \Lambda_i S_{i,j} - q_j G_j) < \nu^{-1}(d-w_j)$ for each j. Having the left-hand side have low degree is the same as the coefficients for $x^{\lceil \nu^{-1}(d-w_j)\rceil},\dots,x^{\deg q_j + \deg G_j}$ should all be zero. That requirement is a system of linear equations in the coefficients of the Λ_i and q_j, so there is a non-zero solution as long as there are more variables than equations.

Now, the number of equations is

$$\sum_{j=1}^{\sigma} \left(\deg q_j + \deg G_j - \lceil \nu^{-1}(d-w_j)\rceil + 1\right)$$

The number of variables is the number of free coefficients at our disposal. Let h be such that $\deg\Lambda_h = \nu^{-1}(d-\eta_i)$. We need to ensure that truly $\deg(\Phi_{\nu,\bar{\boldsymbol{w}}}(\boldsymbol{v})) = d$, which we can do if we can fix Λ_h to have leading coefficient 1. This might not always be possible, depending on the linear system of equations; assume first that it is possible. For the remaining free coefficients, we need to consider for each j whether $\deg G_j < \lceil \nu^{-1}(d-w_j)\rceil$: in this case, then the first few coefficients of $q_j(x)$ has no influence on solving our system, since they affect only lower-terms of

$\sum_{i=1}^{\rho} \Lambda_i S_{i,j} - q_j G_j$. Thus, the number of free variables affecting the system is:

$$\sum_{\substack{i=1 \\ i\neq h}}^{\rho} \lfloor \nu^{-1}(d-\eta_i)+1 \rfloor + \nu^{-1}(d-\eta_h) + \sum_{j=1}^{\sigma} (\deg q_j + 1) - \sum_{j=1}^{\sigma} \operatorname{pos}(\lceil \nu^{-1}(d-w_j) \rceil - \deg G_j)$$

Since we have already secured a non-zero solution, we need only that this number of coefficients is *at least* the number of equations; that means we have a solution within our degree bounds whenever

$$\sum_{j=1}^{\sigma} \operatorname{pos}(\deg G_j - \lceil \nu^{-1}(d-w_j) \rceil) \leq \sum_{i=1}^{\rho} \lfloor \nu^{-1}(d-\eta_i) \rfloor + (\rho - 1)$$

Now we turn to consider when it is possible to choose Λ_h to have leading coefficient 1. In a homogeneous system of linear equations with more variables than unknowns, one can fix some variable x to a non-zero value if and only if $\operatorname{rank} U' = \operatorname{rank} U - 1$, where U is the system matrix and U' is U where the column corresponding to x is removed: for assume that x cannot be fixed to a non-zero value, so $x = 0$ in all solutions. But that means that the solutions to the system of U' are in bijection to those of U; so they have the same solution dimension, implying the statement on the ranks. If we let U be the system matrix corresponding to the system in the coefficients of the Λ_i and q_j for the above value of d, and let x be the variable corresponding to the leading coefficient of Λ_h, then this condition becomes the exception of the proposition. □

For Type 2, then especially the rounding operators make the implicit description difficult to handle analytically. For the important case $\nu = 1$, in an effort to make it slightly more explicit, we give the following corollary:

Corollary 2.40. *Consider a 2D key equation of Type 2 with $\nu = 1$. Define $\zeta_j = \deg G_j + w_j$ for $j = 1, \ldots, \sigma$ and $\zeta_{\sigma+1} = -\infty$, and assume that $\zeta_1 \geq \zeta_2 \geq \ldots \geq \zeta_{\sigma+1}$. Define for $j = 1, \ldots, \sigma$:*

$$d_j = (\rho + j)^{-1} \Big(\sum_{i=1}^{\rho} \eta_i + \sum_{h=1}^{j} (\nu \deg G_h + w_h) - \rho + 1 \Big)$$

Let $\hat{\sigma}$ be the greatest index such that $\zeta_{\hat{\sigma}+1} \leq \lceil d_{\hat{\sigma}} \rceil < \zeta_{\hat{\sigma}}$. Then d of Proposition 2.39 has $d = \lceil d_{\hat{\sigma}} \rceil$. Furthermore, if $d_{\hat{\sigma}}$ is not an integer and under the same assumption on U, U' as in Proposition 2.39, then there exists at least two minimal solutions which differ by more than an $\mathbb{F}$-scaling.

Proof. We will show that $d_{\hat{\sigma}}$ of the corollary equals d of Proposition 2.39. For $\nu = 1$ then d is the least integer satisfying

$$\sum_{j=1}^{\sigma} \operatorname{pos}(\deg G_j - (d - w_j)) \leq \sum_{i=1}^{\rho} (d - \eta_i) + (\rho - 1) \tag{2.7}$$

Note that if the above inequality is sharp, then there must be more than one degree of freedom in the solution space. By the assumption on ordering of the $\deg G_j + w_j$, then there is a maximal h such that $\deg G_j - (d - w_j) \geq 0$, i.e. $\zeta_{h+1} < d \leq \zeta_h$. The terms for $j > h$ in the sum on the left-hand side of (2.7) are then all zero so it reorders to:

$$d \geq (\rho + h)^{-1} \Big(\sum_{i=1}^{\rho} \eta_i + \sum_{j=1}^{h} (\deg G_j + w_j) - (\rho - 1) \Big) \tag{2.8}$$

The right-hand side is exactly d_h. So d must equal the least integer greater than the above right-hand side, so $d = \lceil d_h \rceil$. Now, this d must exist, by Proposition 2.39, and it satisfies that $\zeta_{h+1} < d \leq \zeta_h$, by the definition of h; this shows that the $\hat{\sigma}$ of the corollary is well-defined since at least h is a choice satisfying the restrictions. Conversely, any other valid choice of $\hat{\sigma}$ implies that $d_{\hat{\sigma}}$ satisfies (2.7), and so it follows from the minimality of d and the decreasing order of ζ_j that $h = \hat{\sigma}$.

For the remark on multiple independent solutions, realise that if d is not an integer, then (2.8) must be a sharp inequality and therefore also (2.7); that implies more than one degree of freedom in the linear system of equations which determine the unknowns for the Λ_i and q_j for $j = 1, \ldots, \hat{\sigma}$. □

Remark. The un-pretty exception for the Type 2 case, for which Proposition 2.39 does not hold, looks alien but unfortunately can not be avoided: it is simply the case that for certain degenerate 2D key equations, the smallest solution is surprisingly large. The linear algebraic description should give the impression that for a "random" 2D key equation, this happens quite rarely, though.

As an example, consider a classical key equation, i.e. $\sigma = \rho = 1$, with $S_1(x) = x^{n-1} - 1$ and $G_1(x) = x^n$ for some integer $n > 1$, and trivial weights $\nu = 1$ and $\eta_1 = w_1 = 0$. The matrix to minimise is $M = \left(\begin{smallmatrix} 1 & S_1 \\ 0 & G_1 \end{smallmatrix}\right)$. Applying Mulders–Storjohann, then after only two row operations we get the result $\left(\begin{smallmatrix} x^{n-1}+1 & -1 \\ -x & x \end{smallmatrix}\right)$. I.e. the smallest solution to the key equation has the worst imaginable degree, namely $n - 1$, while by Corollary 2.40, we would expect a solution of degree $\lceil n/2 \rceil$. ◆

Remark. The upper bounds are actually quite similar for both types of 2D key equations in the case $\nu = 1$, since d_j is actually the determinant of the sub-matrix of $\Phi_{\nu,\boldsymbol{w}}(M)$ consisting of the first $\rho + j$ rows and columns.

Some of the complications for Type 2 stem from the fact that if some of the σ congruences have low enough degree, measured in $\deg G_j + w_j$, compared to the other congruences, they will not add complexity to the solution of the 2D key equation; in the sense that the worst-case solution size will not increase. Any solution to only the high-degree key equations will directly extend into one over the low-degree ones.

In Power decoding in Chapter 4, we will see how this is expressed as an upper bound on the "powering" that it makes sense to include in the 2D key equation one wants to solve for decoding: a sought polynomial is a solution to a certain congruence for

any power, and adding powers will exclude other "false" solutions—but only to a certain extend. ♦

Example 2.41. *Using* $(\nu, \boldsymbol{w}) = (1, \mathbf{0})$ *in Example 2.37, then necessarily* $\hat{\sigma} = \sigma = 1$. *Now,* $\det(\Phi_{\nu,\boldsymbol{w}}(M)) = m_{3,3} = x^7 + x^6 + x^5 + x^2$. *Indeed, we saw the degree of the minimal solution was* $2 \leq 3^{-1} \cdot (7 + 1)$.

With $(\nu, \boldsymbol{w}) = (2, (0, 3, 4))$, *then* $\deg \det(\Phi_{\nu,\boldsymbol{w}}(M)) = 21$, *and we saw that the degree of the* $\Phi_{\nu,\boldsymbol{w}}$*-image of the minimal solution was* $7 \leq 3^{-1} \cdot (21 + 1)$. ♠

Remark. For $\sigma = \rho = \nu = 1$ then Mulders–Storjohann applied to $\Phi_{\nu,\bar{\boldsymbol{w}}}(M)$ is exactly the same as running the Extended Euclidean algorithm on $G_1(x)$ and $S_1(x)$ and halting execution on the first iteration i where $\deg v_i + \eta_i < \deg s_i + w_i$, where $s_i(x)$ is the remainder and $v_i(x)$ the coefficient to $S_1(x)$ calculated in the ith iteration of the Euclidean algorithm. ♦

2.5.2 The Demand–Driven algorithm

We will show how Mulders–Storjohann admits a speedup when solving 2D key equations, by the following observation: it is essentially sufficient to keep track of only the first ρ columns of V during the algorithm, and one can then calculate the other entries when the need arise. The result is Algorithm 2, and we will in this section prove its correctness and computational complexity. This algorithm, which we call the Demand–Driven algorithm (D–D), turns out to be faster than Mulders–Storjohann in some cases where ρ is low compared to σ.

For the case $\rho = 1$, the algorithm degenerates into a simpler version: Algorithm 3. The result bears a striking resemblance to the Berlekamp–Massey generalisation for synthesising Multi-sequence Linear-Feedback Shift Registers [SS06], though the manner in which these algorithms are obtained is very different.

Recall the value function ψ of Definition 2.20 on page 19. Overload ψ to $\mathbb{N}_0 \times \{1, \ldots, \rho + \sigma\} \to \mathbb{N}_0$ by $\psi(\theta, h) = (\sigma + \rho)\theta + h$, i.e. for any non-zero $\boldsymbol{v} \in \mathbb{F}[x]^{\rho+\sigma}$, $\psi(\boldsymbol{v}) = \psi(\deg \boldsymbol{v}, \mathrm{LP}(\boldsymbol{v}))$. We will use a helper function $\mathsf{previous}(\theta, h)$ which gives the degree and leading position a vector in $\Phi_{\nu,\bar{\boldsymbol{w}}}(\mathcal{M})$ should have for attaining the greatest possible ψ-value less than $\psi(\theta, h)$:

$$\mathsf{previous}(\theta, h) = \arg\max_{\theta', h'} \{\psi(\theta', h') \mid \psi(\theta', h') < \psi(\theta, h) \ \wedge \ \theta' \equiv \bar{w}_{h'} \mod \nu\}$$

Note that if $\nu = 1$, then we simply have:

$$\mathsf{previous}(\theta, h) = \begin{cases} (\theta, h - 1) & \text{if } h > 1 \\ (\theta - 1, \rho + \sigma) & \text{if } h = 1 \end{cases}$$

We will prove the correctness of the algorithm by showing that the computations correspond to a possible run of a slight variant of Mulders–Storjohann; first we need a technical lemma:

Algorithm 2 Demand-Driven algorithm for 2D key equations

Input: $\Phi_{\nu,\bar{\boldsymbol{w}}}(M) = \left(\begin{array}{c|c} \operatorname{diag}\left(x^{\eta_1},\dots,x^{\eta_\rho}\right) & \tilde{S} \\ \hline 0 & \operatorname{diag}\left(\tilde{G}_1,\dots,\tilde{G}_\sigma\right) \end{array}\right)$.
Let $\tilde{\boldsymbol{m}}_i$ be the rows of this matrix.

Output: The first ρ columns of a basis of $\mathcal{M}$ in $\Phi_{\nu,\bar{\boldsymbol{w}}}$-weighted weak Popov form.

$\Lambda =$ first ρ columns of $\Phi_{\nu,\bar{\boldsymbol{w}}}(M)$
$\left(\theta_i,\alpha_i\right) = \left(\deg(\tilde{\boldsymbol{m}}_i), \operatorname{LC}(\operatorname{LT}(\tilde{\boldsymbol{m}}_i))\right)$ for $i = 1,\dots,\rho+\sigma$
$W = \left\{(i, \operatorname{LP}(\tilde{\boldsymbol{m}}_i)) \mid 1 \le i \le \rho \ \wedge\ \operatorname{LP}(\tilde{\boldsymbol{m}}_i) \neq i\right\}$
while $W \neq \emptyset$ **do**
  $(i,h) = \mathsf{pop}(W)$
  if $\theta_i < \theta_h$ **then** swap $(i,\boldsymbol{\lambda}_i,\alpha_i,\theta_i)$ and $(h,\boldsymbol{\lambda}_h,\alpha_h,\theta_h)$
  $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i - \frac{\alpha_i}{\alpha_h} x^{\theta_i-\theta_h}\boldsymbol{\lambda}_h$
  repeat
    $(\theta_i,h) = \mathsf{previous}(\theta_i,h)$
    $\alpha_i =$ coefficient to x^{θ_i} in $\begin{cases} \lambda_{i,h} & \text{if } h \le \rho \\ \sum_{j=1}^{\rho} \lambda_{i,j}\tilde{S}_{j,h} \mod \tilde{G}_h & \text{otherwise} \end{cases}$
  until $\alpha_i \neq 0$
  if $i \neq h$ **then**
    if $(h,j) \in W$ for some j **then**
      swap $(i,\boldsymbol{\lambda}_i,\alpha_i,\theta_i)$ and $(h,\boldsymbol{\lambda}_h,\alpha_h,\theta_h)$
      replace (h,j) with (i,j) in W.
    else
      $\mathsf{push}(W,(i,h))$
return $\Phi^{-1}_{\nu,\boldsymbol{\eta}}(\Lambda)$

Lemma 2.42. *Consider Algorithm 1 with input $\Phi_{\nu,\bar{\boldsymbol{w}}}(M)$, and let $\tilde{G}_1,\dots,\tilde{G}_\sigma$ be as in Algorithm 2. Consider now a variant of the algorithm where we, when replacing some $\boldsymbol{v}_j$ with $\boldsymbol{v}'_j$ in a row reduction, instead replace it with*

$$\boldsymbol{v}''_j = (v'_{j,1},\dots,v'_{j,\rho},v'_{j,\rho+1} \bmod \tilde{G}_1,\dots,v'_{j,\rho+\sigma} \bmod \tilde{G}_\sigma)$$

This does not change correctness of the algorithm or the upper bound on the number of row reductions performed.

Proof. Correctness follows if we can show that each of the σ modulo reductions could have been achieved by a series of $\mathbb{F}[x^\nu]$ row operations on the current matrix V after the row reduction producing $\boldsymbol{v}'$, since then V would remain a basis of $\Phi_{\nu,\bar{\boldsymbol{w}}}(\mathcal{M})$.

Consider the modulo reduction on the $(h+\rho)$th position. This could be achieved by adding a multiple of the vector $\boldsymbol{g}_h = (0,\dots,0,\tilde{G}_h,0,\dots,0)$, with position $h+\rho$ non-zero, to $\boldsymbol{v}'$. That this multiple is in $\mathbb{F}[x^\nu]$ follows from the fact that any $\boldsymbol{u} \in \Phi_{\nu,\bar{\boldsymbol{w}}}(\mathcal{M})$ has $\deg u_{\rho+h} \equiv w_h \mod \nu$. Since $\boldsymbol{g}_h$ is a row in $\Phi_{\nu,\boldsymbol{w}}(M)$, then as long as this has not yet been row reduced, the $(\rho+h)$th position reduction is allowed. Notice that if $\psi(\boldsymbol{v}') < \psi(\boldsymbol{g}_h)$ then the reduction using $\boldsymbol{g}_h$ is void.

Algorithm 3 Simplified Demand–Driven algorithm for 2D key equations with $\rho = 1$

Input: $\tilde{S}_j \leftarrow x^{w_j} S_{1,j}(x^\nu)$, $\tilde{G}_j \leftarrow x^{w_j} G_j(x^\nu)$ for $j = 1, \ldots, \sigma$
Output: The first column of a basis of $\mathcal{M}$ in $\Phi_{\nu,\bar{\boldsymbol{w}}}$-weighted weak Popov form.

$(\theta, h) \leftarrow (\deg, \text{LP})$ of $(x^{\eta_1}, \tilde{S}_1, \ldots, \tilde{S}_\sigma)$
if $h = 1$ **then return** $(1, 0, \ldots, 0)$
$(\lambda_1, \ldots, \lambda_{\sigma+1}) \leftarrow (x^{\eta_1}, 0, \ldots, 0)$
$\alpha_j x^{\theta_j} \leftarrow$ the leading monomial of $\tilde{G}_{j-1}$ for $j = 2, \ldots, \sigma + 1$
while $\deg \lambda_1 \leq \theta$ **do**
    $\alpha \leftarrow$ coefficient to x^θ in $(\lambda_1 \tilde{S}_{h-1} \mod \tilde{G}_{h-1})$
    if $\alpha \neq 0$ **then**
        if $\theta < \theta_h$ **then** swap $(\lambda_1, \alpha, \theta)$ and $(\lambda_h, \alpha_h, \theta_h)$
        $\lambda_1 \leftarrow \lambda_1 - \frac{\alpha}{\alpha_h} x^{\theta - \theta_h} \lambda_h$
    $(\theta, h) \leftarrow \mathsf{previous}(\theta, h)$
    if $h = 1$ **then** $(\theta, h) \leftarrow \mathsf{previous}(\theta, h)$
return $\left(x^{-\eta_1}\lambda_1, \ldots, x^{-\eta_1}\lambda_{\sigma+1}\right)_{|x=x^{1/\nu}}$

Introduce now a loop invariant involving $J_h = \{\boldsymbol{g}_h\}$, a subset of the current rows in V having two properties: that $\boldsymbol{g}_h$ can be constructed as an $\mathbb{F}[x^\nu]$-linear combination of the rows in J_h; and that each $\boldsymbol{v} \in J_h$ has $\psi(\boldsymbol{v}) \leq \psi(\boldsymbol{g}_h)$. After row reductions on rows not in J_h, the $(\rho + h)$th modulo reduction is therefore allowed, since $\boldsymbol{g}_h$ can be constructed by the rows in J_h. On the other hand, after a row reduction on a row $\boldsymbol{v} \in J_h$ by some $\boldsymbol{v}_k$ resulting in $\boldsymbol{v}'$, the hth modulo reduction has no effect since $\psi(\boldsymbol{v}') < \psi(\boldsymbol{v}) \leq \psi(\boldsymbol{g}_h)$. Afterwards, J_h is updated as $J_h = J_h \setminus \{\boldsymbol{v}\} \cup \{\boldsymbol{v}', \boldsymbol{v}_k\}$ and the loop invariant is kept since $\psi(\boldsymbol{v}_k) \leq \psi(\boldsymbol{v})$.

Since $\psi(\boldsymbol{v}''_j) \leq \psi(\boldsymbol{v}'_j)$ the proof of Theorem 2.22 shows that the number of row reductions performed is not greater than in Algorithm 1. □

We will say for a matrix U that there is a "collision on (i, j)" if $i \neq j, \text{LP}(\boldsymbol{u}_i) = \text{LP}(\boldsymbol{u}_j)$, i.e. one could perform a row reduction involving $\boldsymbol{u}_j$ and $\boldsymbol{u}_i$; we will also say that these rows are "involved in a collision". Now the proof of correctness of the algorithm. The proof is technical but basically just establishes that W is a "worklist" containing indices of rows involved in a collision, which means we should at some point row reduce it. All indices i not thusly in W have been "parked" such that $\text{LP}(\boldsymbol{v}_i) = i$, and they will only be used further in the algorithm if there is a row named in W with leading position i.

Theorem 2.43. *Algorithm 2 is correct.*

Proof. Let V be the matrix continually changing in the variant of Algorithm 1 described in Lemma 2.42. For notational convenience, and simplicity in the description of Algorithm 2, we will consider a further variant of Algorithm 1 where we sometimes swap two rows, to be described momentarily.

Let us write $(i, \star) \notin W$ to mean $\neg\exists j \,.\, (i, j) \in W$, and similarly for $(\star, i) \notin W$. We

will then demonstrate that each iteration of Algorithm 2 corresponds to one row reduction on V, and that the following loop invariants hold:

1. Λ is the first ρ columns of V;
2. $\alpha_i x^{\theta_i}$ is the leading monomial of $\mathrm{LT}(\boldsymbol{v}_i)$;
3. If there is a collision (i,j) then $\exists k\,.\,(i,k) \in W \vee (j,k) \in W$;
4. If $(i,h) \in W$ then $i \neq h$ and $\mathrm{LP}(\boldsymbol{v}_i) = \mathrm{LP}(\boldsymbol{v}_h) = h$;
5. For any i, there is at most one pair in W with first position i.
6. If $(i,\star) \notin W$ then $\mathrm{LP}(\boldsymbol{v}_i) = i$;

The invariants are clearly true after initialisation; assume now they are true on entry of the iteration, and we will show they are true on exit. Once Algorithm 2 terminates, by Invariant 3 then there are no collisions, so V is in weak Popov form, so by Invariant 1 the result is correct.

If $W \neq \emptyset$, then by Invariant 4 there is a collision (i,h), and the considered variant of Algorithm 1 could have chosen to perform a row reduction on this. Note that Invariant 4 also implies that $(\star,i) \notin W$ and $(h,\star) \notin W$, and that Invariant 5 then implies that no other element in W contains i.

Now we possibly perform a swap of rows of V such that the row to be reduced is $\boldsymbol{v}_i$, i.e. if $\deg \boldsymbol{v}_i < \deg \boldsymbol{v}_j$ we swap, which is exactly when $\theta_i < \theta_j$ in Line 6. We note that the Invariants 1 and 2 are maintained since everything involved is swapped. The remaining invariants never distinguish the two swapped rows, which means all non-violations and violations are kept intact. Thus for the remainder of the proof, we can assume that no swap was performed since this will be indistinguishable from the other case.

In Line 7 we perform the actual row reduction, and Invariant 1 is seen to be maintained, due to Invariant 2. The ensuing loop is for maintaining Invariant 2: it will go through possible degrees and leading positions of the updated row $\boldsymbol{v}_i$ in descending order of ψ until it finds one with non-zero coefficient. Note that $\psi(\boldsymbol{v}_i') < \psi(\boldsymbol{v}_i)$, so the loop initiates correctly, and that previous will skip those degree–leading position combinations which are guaranteed to have coefficient zero from the sparsity of polynomials in vectors in $\mathcal{M}$. The calculation in Line 10 is also exactly correct for Lemma 2.42's variant of Algorithm 1.

Refer to V' as V after this row operation, and refer to h' as the updated value of h. The last thing is then to maintain Invariants 3–6. Note that all possible violations must involve i since only for i we have $\boldsymbol{v}_i' \neq \boldsymbol{v}_i$. We distinguish the three cases of the if-statements:

- $i = h'$: Then $\boldsymbol{v}_i'$ cannot be involved in any collision: for if there was a $j \neq i$ such that (j,i) is a collision in V', then $\mathrm{LP}(\boldsymbol{v}_j) = \mathrm{LP}(\boldsymbol{v}_j') = \mathrm{LP}(\boldsymbol{v}_i') = i$ which means that $(j,i) \notin W$ due to Invariant 4 holding on entry of the iteration (at which time $\mathrm{LP}(\boldsymbol{v}_i) \neq i$); but then $(j,\star) \notin W$ which means $\mathrm{LP}(\boldsymbol{v}_j) = j$ by Invariant 6. Thus, we must have $j = h' = i$ but this was false by assumption. Since then $\boldsymbol{v}_i'$

is involved in no collisions, the remaining invariants are seen to hold since: all possible violations to them were regarding i; since i is no longer in any pair of W; and since $\text{LP}(\boldsymbol{v}'_i) = i$.

- $i \neq h'$ and $\exists j \,.\, (h', j) \in W$: That means $(\star, h') \neq W$ by Invariant 4 at iteration entry, and since $(\star, i) \notin W$, we do not create violations to the invariants by swapping $\boldsymbol{v}'_i$ and $\boldsymbol{v}'_{h'}$ when also replacing (h', j) with (i, j) in W. After this swap, we have $\text{LP}(\boldsymbol{v}'_{h'}) = h'$. There now can't be any collision involving $\boldsymbol{v}'_{h'}$: for if there was one (h', k), then $(k, \star) \notin W$ since $(\star, h')$ at iteration entry, and this means by Invariant 6 at iteration entry that $k = \text{LP}(\boldsymbol{v}_k) = \text{LP}(\boldsymbol{v}'_k) = \text{LP}(\boldsymbol{v}'_{h'}) = h$, which is false by assumption. Summarily, since we then have removed the only element in W involving h' and since $\text{LP}(\boldsymbol{v}'_{h'}) = h'$, the invariants are again maintained.
- $i \neq h'$ and $(h', \star) \notin W$: By Invariant 6, we have $\text{LP}(\boldsymbol{v}_{h'}) = h'$ and so (i, h') is a collision, meaning the invariants are maintained by adding (i, h') to W. □

And now the complexity of the algorithm.

Proposition 2.44. *Algorithm 2 has asymptotic complexity*

$$O\big(\rho(\rho+\sigma)(\rho+\sigma+\nu^{-1}\Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)))\tilde{P}\big) \subset O\big(\rho(\rho+\sigma)(\rho\,\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M))+\sigma)\tilde{P}\big)$$

where $\tilde{P} = O(\nu^{-1}\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)))$ *if all* G_j *are powers of* x*, and where* $\tilde{P} = O(\text{P}(\nu^{-1}\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M))))$ *otherwise.*

Proof. Let $P = \rho\tilde{P}$ be an upper bound on the cost of executing Line 10, and we will afterwards show the proposition's claims on $\tilde{P}$. First, since each iteration of Algorithm 2 is exactly a row reduction in Lemma 2.42's variant of Algorithm 1, the number of iterations is most $(\rho+\sigma)(\nu^{-1}\Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) + \rho + \sigma)$ by Theorem 2.22 on page 19. Apart from the repeat-loop, only Line 7 is not $O(1)$, and this has complexity at most $O(\rho\,\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M))) \subset P$.

Each iteration through the repeat loop costs at most P. Each iteration will decrease the upper bound on $\psi(\boldsymbol{v}'_i)$, so by arguments exactly like those of the proof of Theorem 2.22, the total number of times through the repeat loop is then also $m(\nu^{-1}\Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) + m)$. The relaxation is obtained by applying Lemma 2.38 on page 32.

Now for P. $\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M))$ is an upper bound on the degrees of all in-going polynomials, and so trivially $P = O\big(\rho\text{P}(\nu^{-1}\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)))\big)$, remembering again that the polynomials are sparse by a factor ν. Note, however, that if $\tilde{G}_h$ is a power of x, the sought α_i is simply a coefficient in a sum of polynomial products; for this we need not perform the full multiplications, but can perform a convolution for each product for just this one coefficient, and so the computation can be done in the slightly improved $P = O(\rho\nu^{-1}\text{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)))$. □

For the case $\rho = 1$, the worklist W becomes trivially simple so the algorithm degenerates. Since this is an interesting special case (see e.g. Table 2.1), and the

resulting algorithm is arguably much simpler, we have given it as Algorithm 3. The algorithm has exactly the same asymptotic complexity (for $\rho = 1$) as Algorithm 2.

Theorem 2.45. *Algorithm 3 is correct.*

Proof. We will prove the algorithm using the correctness of Algorithm 2, so we reuse the variable names from there. By the structure of $\Phi_{\nu,\bar{\boldsymbol{w}}}(M)$, there is initially either 0 or 1 collisions depending on the leading position of $\tilde{\boldsymbol{m}}_0$. The former case is handled by the fail-fast Line 2. In the other case, W will be initially $\{(1, \operatorname{LP}(\tilde{\boldsymbol{m}}_1))\}$, and each iteration of Algorithm 2, we will either have no more collisions or exactly one, still having row 1 as first position. Thus, we can dispense with W, and h simply contains the second element of this one pair in W.

Since we then always examine the same row, the repeat-loop of Algorithm 2 can be merged with the main loop, which means that the "search" for the new leading position and degree becomes a guard around the lines actually performing the row reduction. The algorithm is finished as soon as W is not re-filled with $(1, \operatorname{LP}(\tilde{\boldsymbol{m}}_1))$ which happens exactly when $\operatorname{LP}(\tilde{\boldsymbol{m}}_1) = 1$, i.e. $\deg \lambda_1 > \theta$, when θ is always updated with previous and skips leading positions of 1. □

Remark. For $\sigma = \rho = 1$ and $G_1(x) = x^d$, a 2D key equation is simply a classical key equation such as (2.2). Here Algorithm 3 degenerates to exactly the computations of the Berlekamp–Massey algorithm [Ber68, Mas69]. Compare this with the remark on page 36 on the Mulders–Storjohann algorithm being a generalisation of the Extended Euclidean algorithm. ◆

2.6 Summary of complexities

Tables 2.2–2.4 provide an overview of the long list of methods given in this chapter and their convoluted complexity expressions. We will refer back to these repeatedly in the remainder of the thesis. We have provided the complexity estimates in their specific, unrelaxed form, in particular using both the orthogonality defect and max-degree; this slightly obscures an overview of the relative speed of the methods, but is necessary since the parameters we will encounter in applications turn out to have a variety of relative sizes. In each table, we have reused the parameter names from the sections pertaining to this case, but we have also introduced short-hands described in the captions.

2.7 Related work

The type of module minimisation we are considering here could be accomplished by Buchberger's algorithm, which can compute a Gröbner basis for any module with regards to any monomial ordering; however, without further arguments, the

Complexity for obtaining a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ in weak Popov form

Algorithm	Complexity	Page	Assumptions
Mulders–Storjohann	$m^2(\delta+m)\gamma$	20	
Alekhnovich	$\mathrm{M}(m)(\mathrm{P}(\delta)\log(\delta)+\mathrm{P}(\gamma))$	23	$m \in O(\delta)$
GJV	$\mathrm{M}(m)\mathrm{P}(\nu\gamma)\log(m\nu\gamma)^{O(1)}$	25	

Table 2.2: We have let $\gamma = \nu^{-1}\mathrm{maxdeg}\,(\Phi_{\nu,\boldsymbol{w}}(V))$ and $\delta = \nu^{-1}\Delta(\Phi_{\nu,\boldsymbol{w}}(V)) \leq m\gamma$;

Complexity for computing one row in a basis of $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ in weak Popov form

Algorithm	Complexity	Page	Assumptions
Mulders–Storjohann	$m^2(\delta+m)\gamma$	19	
Alekhnovich	$\mathrm{M}(m)\mathrm{P}(\delta)\log(\delta)+m^2\mathrm{P}(\gamma)$	23	$m \in O(\nu\delta)$
GJV	$\mathrm{M}(m)\mathrm{P}(\nu\gamma)\log(m\nu\gamma)^{O(1)}$	25	

Table 2.3: We have let $\gamma = \nu^{-1}\mathrm{maxdeg}\,(\Phi_{\nu,\boldsymbol{w}}(V))$ and $\delta = \nu^{-1}\Delta(\Phi_{\nu,\boldsymbol{w}}(V)) \leq m\gamma$;

Complexity for finding a solution to a 2D key equation

Algorithm	Complexity	Page	Assumptions
Mulders–Storjohann	$(\rho+\sigma)^2(\delta+\rho+\sigma)\gamma$	19	
Alekhnovich	$\mathrm{M}(\rho+\sigma)\mathrm{P}(\delta)\log(\delta)+(\rho+\sigma)^2\mathrm{P}(\gamma)$	23	$\rho,\sigma \in O(\nu\delta)$
GJV	$\mathrm{M}(\rho+\sigma)\mathrm{P}(\nu\gamma)\log\big((\rho+\sigma)\nu\gamma\big)^{O(1)}$	25	
Demand–Driven	$\rho(\rho+\sigma)(\delta+\rho+\sigma)\tilde{P}$	40	

Table 2.4: We have let $\gamma = \nu^{-1}\mathrm{maxdeg}\,(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) = \nu^{-1}\max\{\eta_1,\ldots,\eta_\rho,\nu\deg G_1 + w_1,\ldots,\nu\deg G_\sigma + w_\sigma\}$ and $\delta = \nu^{-1}\Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) < \rho\gamma$. Recall that $\tilde{P} = O(\mathrm{P}(\gamma))$ in general, while if all G_j are powers of x, then $\tilde{P} = O(\gamma)$.

running time of Buchberger's algorithm is doubly-exponential in the max-degree of the initial basis!

The earliest result on module minimisation specifically for $\mathbb{F}[x]$ that I have found is Lenstra [Len85], where a minimisation algorithm is developed with the aim of factoring multivariate polynomials. The algorithm runs in $O\big(m^3(\delta+1)\gamma\big)$ using the notation of Table 2.2 on page 42, and so is strictly slower than Mulders–Storjohann, though. Lenstra introduced the orthogonality defect for complexity analysis, and it is here I found the inspiration for also describing the complexity of the Mulders–Storjohann and Alekhnovich algorithms similarly; it was a stroke of fortune that this formulation turned out to have applications in various modules cropping up in decoding algorithms.

Currently, the GJV is the fastest known algorithm for computing bases in weak Popov form, for general initial bases and measured in their max-degree. It would seem that asymptotically, this result can only be marginally improved since it is essentially the complexity of multiplying together matrices of size as the initial basis; further details are in the introduction of [GJV03]. The GJV is a D&C adaption of a previous algorithm by Beckermann and Labahn [BL92]. As remarked in Section 2.5, module minimisation and approximation (as d-approximants or Padé) are obviously computationally intimately connected.

As demonstrated in Table 2.1, the 2D key equation generalises a list of previously studied approximants and "key equations". One of the original aims with the design of 2D key equations was to encompass elegantly a wide range of applications in coding theory, as we will demonstrate in the remainder of the thesis. It was a consequence of the underlying mathematics that the types of Padé approximations were captured as well. Indeed, one sees from Table 2.1 that apart from the "Type", each Padé approximation is nearly also a certain type of key equation.

The idea of using module minimisation, or lattice reduction methods, to solve Padé type approximations is not new. For instance, Lenstra, Lenstra and Lovász in their famous paper on basis reduction over $\mathbb{Z}$ describe how to solve Diophantine approximation problems using their algorithm, and using an initial basis essentially like that of (2.3) on page 30 for $\rho = 1$. Another example is the rational reconstruction problem (over $\mathbb{F}[x]$) in [OS06] which is solved by a Simultaneous Padé approximation using module minimisation.

Massey [Mas69] described the classical key equation for Reed-Solomon codes [Ber68] as a Linear Feedback-Shift Register (LFSR), and solved these using the (thereof named) Berlekamp–Massey algorithm. It has also long been known that this key equation is a Padé approximation, as mentioned in e.g. [Fit95]. I thus find it slightly surprising that I have found no mention of the near-equivalence between the classical notion of Simultaneous Padé approximation [BGM96] and Multi-sequence LFSR [FT91], despite a tremendous amount of attention devoted to the latter, especially since the inception of Power decoding, e.g. [SS06, Wang^3, SS11, ZW11].

Sugiyama et al. [SKHN75] gave a key equation for Goppa codes and solved this using the extended Euclidean algorithm; since $G(x)$ is not a power of x in this case, the Berlekamp–Massey can't immediately solve it. However, using the trick of Patterson [Pat75], one *can* rewrite the key equation into one using x^d; we will see similar tricks in both Section 3.4 and Section 4.2. This naturally raised the question of what relation there is between the Berlekamp–Massey algorithm and the Euclidean algorithm when $G(x) = x^d$. That has been expounded in several publications, e.g. [Dor87, HJ00], but the most elegant connection was given by Fitzpatrick [Fit95], who introduced the Gröbner basis view for these simple key equations.

A Multi-sequence LFSR is a 2D key equation with $\rho = 1$ and $G_j = x^{d_j}$ for some d_j, as well as trivial weights $\nu = 1, \eta_1 = w_j = 0$. Feng and Tzeng solved these using a generalisation of the Berlekamp–Massey algorithm [FT91], but there was a subtle error when the d_j were not all equal, pointed out and corrected by Schmidt and Sidorenko [SS06, SS11]. I think it speaks of the inherent convolutedness of this algorithm that it took 15 years to point out this flaw, and more than 5 full pages to prove the amendment's correctness [SS11]. The resulting algorithm has complexity $O(\sigma d_{\max}^2)$ with $d_{\max} = \max\{d_j\}$. It also admits a D&C speedup [SB11], yielding a complexity of $O(\mathrm{M}(\sigma) d_{\max} \log(d_{\max}))$. These complexities are also achieved by the Demand–Driven and Alekhnovich algorithm respectively for the corresponding 2D key equation. In fact, I was seeking to achieve the former complexity when I looked for the Demand–Driven speedup of the Demand–Driven algorithm.

Feng and Tzeng also solved Multi-sequence LFSRs using an algorithm generalising the extended Euclidean algorithm [FT89]. As mentioned above, the original extended Euclidean algorithm is essentially equivalent to the Berlekamp–Massey for the case where one argument is a power of x. I think it is particularly intriguing then, that Storjohann–Mulders obviously generalise the Euclidean algorithm, while the Demand–Driven variant Algorithm 2 bears striking similarity to the corrected Feng–Tzeng generalisation of the Berlekamp–Massey when $\rho = 1$ and the G_j are powers of x. I am sure that Fitzpatrick's arguments for the "equivalence" of the two algorithms generalise, though I am not sure what the purpose of such an exercise would be.

Wang, Wang and Wang used $\mathbb{F}[x]$ lattices to solve Multi-sequence LFSRs [Wang^3], and in a number of ways the approach is quite similar to Mulders–Storjohann applied to these special types of 2D key equations. They even give a variant which could possibly be like the Demand–Driven algorithm. For technical reasons, which could probably be overcome, the approach does not directly generalise to either general G_j or general ρ; for doing this one would probably want to introduce the flexible language of weak Popov form and Gröbner bases, and the result would most likely be exactly Mulders–Storjohann.

The key equation has been generalised in another direction by Roth and Ruckenstein [RR00] for finding interpolation polynomials in the Sudan algorithm (discussed in detail in Section 3.4.1); these correspond to a 2D key equation of Type 1 with

$\sigma = 1$ and $G_1 = x^d$ as well as $\nu = 1$. Once again, this is nearly equivalent to a well-known class of approximations: Hermite Padé approximations, which are widely studied [Her78, BGM96]. The algorithm of Roth and Ruckenstein is yet another generalisation of the Berlekamp–Massey and has running time $O(\sum_j (d-\eta_j) \sum_j (N-\eta_j)) \subset O(\rho^2 N d)$; here Mulders–Storjohann has the slower $O(\rho^2 d \sum_j (d-\eta_j))$.[2] When viewed the right way, their algorithm is basically a special case of the Demand–Driven algorithm, and the speedup is achieved by a clever choice and ordering of the row reductions. This means that the polynomials in the first ρ columns will only ever increase in degree, and in such a way that the number of row reductions necessary is reduced by essentially a factor ρ. Furthermore, it means that for finding only *solutions* one can forget about rows as soon as they contain a polynomial in the first ρ positions with degree at least N, since they can never contribute to solutions; thus all remaining row reduction can be counted cheaper. This analysis seem to be completely possible to carry out in full for the Demand–Driven algorithm for this special case, so it seems that it actually *does* run in the same complexity; it would be even more interesting to examine more closely whether this observation generalises to a larger class or even to all 2D key equations.

A further generalisation of the Roth–Ruckenstein approach to the Guruswami–Sudan algorithm was given by Zeh, Gentner and Augot [ZGA11]; they get 2D key equations of Type 1 with $G_j = x^{d_j}$ and $\nu = 1$. Their algorithm is a specialisation of the Fundamental Iterative Algorithm by Feng and Tzeng [FT91] and has running time $O(\rho\sigma^2 d_{\max}^2)$ when one considers $\sigma < \rho \ll d_{\max}$ (as they do); under those assumptions both Mulders–Storjohann and the Demand–Driven algorithm has $O(\rho^3 d_{\max}^2)$ which is slightly worse. Again, one could suspect that observations like those described above could improve the running time of the Demand–Driven algorithm. I am not aware of a D&C variant of this or the Roth–Ruckenstein algorithm, and under the considered assumptions, the GJV algorithm is faster than the above, having $O(\mathrm{M}(\rho)\mathrm{P}(d_{\max}) \log(d_{\max})^{O(1)})$. We will further discuss these cases specialised to decoding Reed–Solomon codes in Section 3.4.

Module minimisation for finding the interpolation polynomial in Guruswami–Sudan was introduced by Lee and O'Sullivan [LO08]. They developed their own algorithm for this, inspired by Buchberger's algorithm for general Gröbner basis computation, and the result is essentially a reinvention of Mulders–Storjohann. They use Gröbner bases for easing certain discussions, but in concluding remarks they attempt to distance themselves from lattice reduction and in particular the Alekhnovich algorithm; in light of this chapter, I find this remark misinformed.

Sakata generalised the original Berlekamp–Massey for multi-dimensional LFSRs. The description uses Gröbner bases for the solution formulation, but I have not in depth investigated the algorithm's similarity to those presented in this chapter; however, in Section 4.4 we will show how to "unfold" a 2-dimensional LFSR into a

[2] Actually, one can easily get $O(\rho(\sum_j d - \eta_j)^2)$ if one properly counts the price for a row reduction. In [RR00], they report a complexity specifically for their application and also employ relations they have between ρ, N and the η_j.

2D key equation, and thus solving it using the framework presented here.

Shortly before I finished this thesis, I was made aware of the work of O'Keeffe [O'K03], who during his PhD under the advice of Fitzpatrick, described a Gröbner basis algorithm for solving a multivariate generalisation of a 2D key equation; the algorithm is non-D&C and seems to be slower than the Mulders–Storjohann, but I have not analysed this special case of the algorithm well enough to be certain. O'Keeffe described various applications, such as solving the Roth–Ruckenstein equation for Sudan decoding (see below and in Section 3.4.1) and finding an interpolation polynomial in Guruswami–Sudan, but precise complexity analyses are missing, and judging from the general behaviour of his algorithm, it seems to be slower than the various solutions we propose in Section 3.1. How exactly the algorithm of O'Keeffe relates to Sakata's is not described either.

Chapter 3

Guruswami–Sudan

The next three chapters will be concerned with decoding, primarily of GRS codes. We will draw heavily on module minimisation and solving of 2D key equations described in the preceding chapter. Our first decoding paradigm will be the Guruswami–Sudan algorithm; this was also historically the first polynomial-time algorithm for decoding GRS codes beyond half the minimum distance, described in [GS99] and building upon the simpler Sudan's algorithm from [Sud97].

We will begin in Section 3.1 by deriving the algorithm as well as the implicit conditions on its parameters, followed by a detailed analysis for turning these conditions into explicit formulas. The algorithm has two expensive steps: interpolation and root-finding. The remainder of the chapter focus on the interpolation step, and we only briefly discuss the root-finding step in Section 3.1. In Section 3.2 we describe a fast interpolation method building directly on module minimisation. We exploit a structural property of this method in Section 3.3 to arrive at an interpolation algorithm which also allows a variation of the list-decoding paradigm, namely maximum-likelihood list decoding. In Section 3.4 we describe how one can alternatively perform the interpolation step by solving a certain 2D key equation. The Guruswami–Sudan algorithm for Hermitian codes, a class of Algebraic Geometric codes, is considered in Section 3.5, with a focus on how to perform the interpolation step fast using a method analogous to that of Section 3.2.

In the remainder of the thesis, we will most of the time be considering a particular $[n,\ k,\ d]$ GRS code, with evaluation points $\alpha_1, \ldots, \alpha_n$ and column multipliers $\beta_1, \ldots, \beta_n$. We will let $\boldsymbol{c} \in \mathcal{C}$ be some sent codeword coming from evaluating some polynomial f, i.e. $\boldsymbol{c} = \mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f) = (\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n))$, and $\boldsymbol{r} = (r_1, \ldots, r_n) =$

$\boldsymbol{c}+\boldsymbol{e}$ will be the received word with some error $\boldsymbol{e}$ on it. Define also $\boldsymbol{r}' = (r'_1, \dots, r'_n) = (r_1/\beta_1, \dots, r_n/\beta_n)$ as a useful shorthand. We let $\mathcal{E} = \{i \mid e_i \neq 0\}$ be the error positions. We will assume $\boldsymbol{r} \notin \mathcal{C}$ to discount trivial cases.

Contributions

- A new geometric proof of the decoding radius of the Guruswami–Sudan algorithm. This reveals quite clearly certain asymptotic properties of the parameter choices, for example Corollary 3.9.
- Good closed-form parameter choices of the Guruswami–Sudan algorithm, and a more precise description of the asymptotic behaviour of possible parameter choices. In particular, we show how the closed form of the list size is very close to the improved Johnson bound by Cassuto and Bruck [CB04].
- More complete discussion of using module minimisation for interpolation than previous work, and our work emulates several known interpolation methods e.g. that of [LO08, BB10, CH10]; see also Section 3.6.
- The step-wise interpolation and multi-trial algorithm. It was developed together with Alexander Zeh and appeared in equivalent form in [NZ13], though its complexity has been analysed here for use with any of the module minimisation algorithms of Chapter 2.
- Solving Q-finding 2D key equations of Zeh, Genter and Augot [ZGA11] using module minimisation; this specialises to the key equations of Roth–Ruckenstein. By module minimising with the Alekhnovich or GJV, we achieve quasi-linear dependence on n, which has not been done for this approach before.
- Using module minimisation for the interpolation step for Guruswami–Sudan decoding Hermitian codes has been suggested before [LO09, Bra10], but we suggest an alternative weighing of the module, and by using the GJV this results in a better complexity than any previous method. Our general analysis of the performance of Mulders–Storjohann and Alekhnovich algorithms also improves the complexity estimates for the two mentioned previous works.

3.1 The main theorem

Definition 3.1. A multivariate polynomial $P \in \mathbb{F}[x_1, \dots, x_\kappa]$ has a zero $(\gamma_1, \dots, \gamma_\kappa) \in \mathbb{F}^\kappa$ with multiplicity s if and only if $P(x_1 + \gamma_1, \dots, x_\kappa + \gamma_\kappa)$ can be written as a linear combination of monomials, each having degree at least s.

Theorem 3.2 (Guruswami–Sudan for GRS codes [GS99, Sud97])**.** *Let $s, \ell, \tau \in \mathbb{Z}_+$ be given. If $Q \in \mathbb{F}[x, y]$ is a non-zero bivariate polynomial of y-degree at most ℓ satisfying*

1. *$Q(x, y)$ has a zero at (α_i, r'_i) with multiplicity s for $i = 1, \dots, n$.*

2. $\deg_{1,k-1} Q < s(n-\tau)$.

and if $|\mathcal{E}| \leq \tau$ *then* $(y-f) \mid Q$.

Proof. Considering $Q \in \mathbb{F}[x][y]$, then the theorem merely states that f is a y-root of Q. Now, $Q(x,f) \in F[x]$ has degree at most $s(n-\tau)$ since $\deg f \leq k-1$, and it has an s-multiple root α_i whenever $f(\alpha_i) = r'_i$, i.e. $i \notin \mathcal{E}$. Thus, $Q(x,f)$ has more roots that its degree so it must be the zero polynomial. □

The theorem states nothing on when such a Q exists, however. Since τ is the only parameter one is interested in as an "end-user" of the above theorem, s and ℓ can be considered parameters that one should choose carefully for ensuring the existence of a Q for a given τ, while minimising computational effort in finding Q and performing root-finding on it. Unsurprisingly, the computational effort increases with increasing s and ℓ, so if there is a choice for given n, k and τ, one prefers small values of s and ℓ. Due to its role, s is often called *the multiplicity* of the theorem or algorithm. Similarly, ℓ is called *the list size*: since there can be at most $\deg_y Q = \ell$ different y-roots, ℓ is an upper bound on the number of codewords within distance τ of r, see also Algorithm 4.

The main characterisation of when a Q exists is easy to obtain:

Definition 3.3. Let the *GS satisfiability function* $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau)$ be given by

$$E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) = (\ell+1)s(n-\tau) - \tbinom{\ell+1}{2}(k-1) - \tbinom{s+1}{2}n$$

Proposition 3.4. *In the context of Theorem 3.2, a* Q *exists if* $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$.

Proof. The first requirement on Q can be given as a list of linear, homogeneous equations in the coefficients of Q, while the second simply states how many coefficients we have at our disposal. Whenever the latter is greater than the former, there must be a non-zero solution.

By Definition 3.1, for each i stating that Q should have a zero (α_i, r'_i) with multiplicity s specifies a homogeneous equation on the coefficient of $Q(\alpha_i + x, r'_i + y)$ for each monomial $x^j y^h$ for $j + h < s$, i.e. $\binom{s+1}{2}$ equations for each i.

Writing $Q = \sum_{t=0}^{\ell} Q_t(x) y^t$ the second requirement is $\deg Q_t < s(n-\tau) - t(k-1)$, so the number of coefficients at our disposal is

$$\sum_{t=0}^{\ell} \big(s(n-\tau) - t(k-1)\big) = (\ell+1)s(n-\tau) - \tbinom{\ell+1}{2}(k-1)$$

Thus the degrees of freedom in the resulting linear system of equations specifying the requirements on Q is exactly $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau)$, whenever this is non-negative. □

For a given code, and for given values of the parameters s and ℓ, there is a maximal τ such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$; denote this value by $\tau(s,\ell)$. By insertion we see that $\tau(1,1) = \lfloor \frac{n-k}{2} \rfloor$, i.e. roughly half the minimum distance.

The following example code will be used throughout the thesis.

Example 3.5. *Consider an* $[250, 70, 181]$ *code over* $\mathbb{F}_{251}$*. We can minimum-distance decode up to* $\tau(1,1) = 90$*. Now, for each* $\tau = 91, \dots, 118$ *there exists* s, ℓ *such that* $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$*, meaning that the decoding radius of the Guruswami–Sudan algorithm for this code is* 118*. With* $(s,\ell) = (1,2)$ *we can choose* τ *as high as* 97*, while all higher values of* τ *requires greater* s *and* ℓ*. For instance,* $\tau = 105$ *requires at least* $(s,\ell) = (2,4)$ *while* $\tau = 118$ *requires* $(s,\ell) = (47,89)$*.* ♠

Proposition 3.4 immediately gives a naïve way to find such a Q: solve the resulting linear system of equations in the coefficients. With no further analysis, this can be done in complexity $O(\mathrm{M}(s^2 n)) \subset O(s^6 n^3)$ [Str69] (assuming $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau)$ is close to 0). As we will see repeatedly in this chapter, we can do much better.

Algorithm 4 Guruswami–Sudan list decoding of GRS codes

Input: The received word $\boldsymbol{r} = (r_1, \dots, r_n)$. Parameters τ, s, ℓ such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ for the given code.

Output: A list of all $f \in \mathbb{F}[x]$ with $\deg f < k$ such that $\mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f) \in \mathcal{C}$ and $\mathrm{dist}(\mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f), \boldsymbol{r}) \leq \tau$.

1 Compute $Q \in \mathbb{F}[x,y]$ such that Q satisfies the requirements of Theorem 3.2.

2 Find all $y - f$ dividing Q such that $\deg f < k$.

3 Return all of these f which satisfy $\mathrm{dist}(\mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f), \boldsymbol{r}) \leq \tau$.

From Theorem 3.2 to a complete decoding algorithm is easy, and we have given such a description as Algorithm 4. If choosing $s = 1$ in Theorem 3.2 and Algorithm 4, then they are usually referred to simply as "Sudan's theorem" and "Sudan decoding" respectively, since that was first described in [Sud97]. In Chapter 4 we will see an alternative decoding algorithm for GRS codes which has several relations with Sudan decoding.

Both Line 1 and Line 2 are non-trivial to compute. We just gave a naïve solution to the former, and we will give several faster alternatives in the remainder of the chapter. For the latter, however, we will devote little time. It turns out that—asymptotically, at least—this root-finding problem is easier than constructing Q, by several orders of magnitude in computational complexity. Proposition 3.6 sums up the best, currently known result.

Proposition 3.6. *Given* $Q \in \mathbb{F}[x][y]$ *with* $\deg_y Q = \ell$ *and* $\deg_x Q = N$*, there exists an algorithm for finding all* y*-roots of* Q *in complexity* $O(\ell^2 \mathrm{P}(N) \log N)$*, assuming that* $\ell, q \in O(N)$ *where* q *is the cardinality of* $\mathbb{F}$*.*

Proof. (sketch) The root-finding method of Roth and Ruckenstein [RR00] (and also nicely described in [Rot06, Section 9.7]) is a relatively fast and elegant way of solving the problem. It has computational complexity $O\big(K(\ell N \log(\ell)^2 + F(\ell))\big)$, where $F(T)$ is the cost of root-finding an $F[x]$ polynomial of degree T, and one seeks only y-roots of Q of degree at most K. We can choose e.g [vzGG03, Theorem

14.14] to have $F(T) = O(T\mathrm{P}(T)\log(qT))$, and so get a decent complexity.

However, in the cases of our interest, K, N are of comparable size and greater than ℓ, so we would like something faster than quadratic in these two. Alekhnovich [Ale05] described a D&C speedup to the Roth–Ruckenstein method to give it complexity $O(\ell^{O(1)}\mathrm{P}(N)\log(N))$. His analysis was a bit asymptotically loose, though, and can easily be improved: in the context of his proof, set $f(1,\ell) = F(\ell)$. The non-recursive cost of $f(t,\ell)$, i.e. the term $\ell^{O(1)}t$, can be improved to $\ell^2\mathrm{P}(t)$, as an upper bound cost of the ℓ different calculations of the shifts $Q(x, y_i + x^{d_i}\hat{y})$. Now the recursive bound has the improved solution $f(t,\ell) \in O(\ell^2\mathrm{P}(t)\log t + t\ell\mathrm{P}(\ell)\log(q\ell))$. The total cost is $f(N,\ell)$ so assuming $q \in O(N)$ we get the sought. □

Remark. Note that the saving from knowing an upper bound on the degree of the y-roots is lost in the D&C algorithm compared to the original Roth–Ruckenstein; it is unclear whether better stopping criteria and better analysis could restore this. ◆

Remark. The Roth–Ruckenstein algorithm, as well as the D&C speedup, actually does more: it finds all $p \in \mathbb{F}[x]$ such that $Q(x,p) \equiv 0 \mod x^m$ for any desired m. By choosing m high enough, all actual roots will be in this list, but the list will also contain certain "spurious roots" which are discarded. The spurious roots, however, will be important in Chapter 5, where we see that some of these actually constitute power series expansions of rational expressions $\frac{p_1}{p_2} \in \mathbb{F}(x)$ such that $Q(x, \frac{p_1}{p_2}) = 0$. ◆

3.1.1 Decoding radius

To determine the decoding radius of the Guruswami–Sudan algorithm, we need to find for which τ we can select $s, \ell \in \mathbb{Z}_+$ such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$. We will actually do this twice: first using a geometric argument, which focus on the asymptotics of $\frac{\ell}{s}$, and second using a purely algebraic argument where we find good, closed expressions for how to choose s and ℓ for any given τ at most the decoding radius. The former is useful for philosophising on the nature of the parameters, and the latter is more practical and important for complexity analysis. We will begin by pruning half the search space:

Lemma 3.7. *For s, ℓ, τ with $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ and $d/2 < \tau \leq d$, then $s \leq \ell$.*

Proof. Assume that $s \geq \ell$. Recall that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) = -\binom{s+1}{2}n + \sum_{t=0}^{\ell} s(n-\tau) - t(k-1)$. Since $s(n-\tau) - t(k-1) \geq 0$ for all $t \leq s$ since $\tau \leq d$, then

$$\sum_{t=0}^{\ell} s(n-\tau) - t(k-1) \leq \sum_{t=0}^{s} s(n-\tau) - t(k-1) = (s+1)s(n-\tau) - \binom{s+1}{2}(k-1)$$

That is to say, $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) \leq E_{\mathrm{GS}}^{[n,k]}(s,s,\tau)$. But $E_{\mathrm{GS}}^{[n,k]}(s,s,\tau) > 0$ implies:

$$(s+1)s(n-\tau) - \binom{s+1}{2}(k-1) - \binom{s+1}{2}n > 0 \qquad \iff \qquad \tau < d/2 \qquad \square$$

Proposition 3.8. *There exists suitable choices for $s \leq \ell$ such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ for any $\tau < n - \sqrt{n(n-d)}$.*

Proof. We will focus on the ratio ℓ/s, so let $\varphi = \ell/s$ be some rational number. Then $E_{\mathrm{GS}}^{[n,k]}(s,\varphi s,\tau) > 0$ becomes

$$\begin{aligned}(\varphi s+1)s(n-\tau) - \tfrac{1}{2}\varphi s(\varphi s+1)(k-1) &> \tfrac{1}{2}ns(s+1) \iff \\ n - \tau - \tfrac{1}{2}\varphi(k-1) &> \tfrac{1}{2}n\frac{1+\frac{1}{s}}{\varphi+\frac{1}{s}}\end{aligned}$$

When $\varphi \geq 1$, we can find an s large enough to satisfy the above if and only if

$$\begin{aligned}n - \tau - \tfrac{1}{2}\varphi(k-1) > \tfrac{1}{2}n\varphi^{-1} &\iff \\ \tau < (1-\tfrac{1}{2}\varphi^{-1})n - \tfrac{1}{2}\varphi(k-1) &\end{aligned} \tag{3.1}$$

The following geometrical argument is exemplified on Figure 3.1 on page 53: whenever $\varphi \geq 1$, the line $\psi(k_\circ) = (1-\frac{1}{2}\varphi^{-1})n - \frac{1}{2}\varphi k_\circ$ is the tangent line of $\Psi(k_\circ) = n - \sqrt{nk_\circ}$ touching at $k_\circ = n\varphi^{-2}$. Since $\Psi(k_\circ)$ is convex for $k_\circ > 0$, then for any $\tau < n - \sqrt{n(k-1)}$, we can choose a $\varphi \geq 1$ such that τ is below the corresponding tangent line of $\Psi(k_\circ)$, so it satisfies (3.1) and therefore some large enough s will satisfy $E_{\mathrm{GS}}^{[n,k]}(s,\varphi s,\tau) > 0$. It will always be sufficient to choose φ such that the corresponding tangent line intersects $\Psi(k_\circ)$ at $k_\circ = k-1$, namely $\varphi = \sqrt{\frac{n}{k-1}}$, but there will be a range of possible values for φ depending on τ. □

Remark. The function $J(n,d) = n - \sqrt{n(n-d)}$ is called the asymptotic Johnson radius: Johnson [Joh62] gave a bound for the size of constant-weight codes, and this can be used to prove that *any* linear code has only a constant number of codewords within any Hamming ball of radius less than $J(n,d)$, where n is the length and d the minimum distance of the code [Gur07].

It is beautiful that the Guruswami–Sudan algorithm can decode exactly up to a long-known bound. We know that there exists infinite families of codes such that there are exponentially many (in n) codewords in bigger balls than $J(n,d)$ [GRS00], but it is unknown whether this is the case for Reed–Solomon codes. See also discussion in Section 3.6. ◆

The following corollary describes the behaviour when decoding toward the limit; due to integer rounding, an integer decoding radius can not "go towards" some limit, so it should be understood in the sense of *real* values of τ, s and ℓ.

Corollary 3.9. *For $\tau \to n - \sqrt{n(n-d)}$, choices of s and ℓ such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ must satisfy $\ell, s \to \infty$ with $\ell/s \to \sqrt{\frac{n}{k-1}}$.*

Proof. Referring to the proof of Proposition 3.8, we see that since $\Psi(k_\circ)$ is convex, then if we let $\tau \to \Psi(k-1)$ we are forced to choosing a tangent of $\Psi(k_\circ)$ ever closer to the tangent passing through $k_\circ = k-1$. In the limit, this yields the given value of φ. For the asymptotic behaviour of s, go a few steps back in the above proof and note

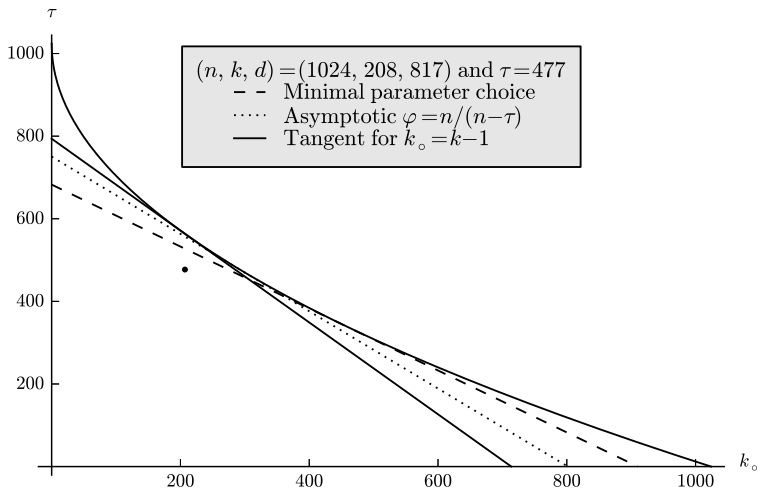

Figure 3.1: An illustration of the geometric proof of Proposition 3.8, using specific parameters. For any tangent to $\Psi(k_\circ)$ for which $(k-1,\tau) = (207, 477)$ is below, corresponds a value φ for which a large enough value of s exists such that $E_{\mathrm{GS}}^{[n,k]}(s, \varphi s, \tau) > 0$. Proposition 3.11 gives $(s, \ell) = (2, 3)$ and these are also the minimal possible. The corresponding tangent is different from the tangent touching $\Psi(k_\circ)$ in $k_\circ = 207$. We also see the tangent corresponding to the value of $\varphi = n/(n-\tau)$; Corollary 3.14 gives upper bounds for the parameter choices given by Proposition 3.11, and the fraction ℓ/s of these bounds will asymptotically approach $n/(n-\tau)$ for n tending to infinity, keeping τ/n and d/n constant.

that we can have $E_{\mathrm{GS}}^{[n,k]}(s, \varphi s, \tau) > 0$ only whenever $\tau < n - \frac{1}{2}\varphi(k-1) - \frac{1}{2}n\frac{1+1/s}{\varphi+1/s}$. Since the right-hand side increases with increasing s, and only in the limit of infinite s do we reach the tangent line of $\Psi(k_\circ)$, and since in the limit of $\tau \to \Psi(k_\circ)$ there is only this one possible tangent line to choose, we must require $s \to \infty$ to get arbitrarily close to this tangent. □

Example 3.10. *Consider again the* $[250, 70, 181]$ *code from Example 3.5. We recall that the decoding radius was 118, and indeed we have* $n - \sqrt{n(n-d)} \approx 118.66$. ♠

3.1.2 Choosing the parameters

Now for some concrete expressions on how to choose s and ℓ. We introduce some short-hands; these ease the derivation, especially if the reader wish to verify the (tedious) calculations of the following proposition, but they also serve to highlight a connection with rational interpolation as done in Chapter 5, especially Proposition 5.7 on page 131.

Proposition 3.11. *For a given code with* $k > 1$ *and a given decoding radius*

$\tau < n - \sqrt{n(n-d)}$, *then* $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ *if* s, ℓ *are chosen as*

$$s = \lfloor s_{\min} + 1 \rfloor \qquad \ell = \left\lfloor \tfrac{\bar{\tau}}{k_\circ} s + \tfrac{1}{2} - \tfrac{\sqrt{D}}{k_\circ} \right\rfloor$$

where

$$s_{\min} = \frac{\tau k_\circ}{\bar{\tau}^2 - n k_\circ} \qquad \bar{\tau} = n - \tau$$
$$D = (s - s_{\min})(\bar{\tau}^2 - n k_\circ) s + \tfrac{k_\circ^2}{4} \qquad k_\circ = k - 1$$

Proof. We seek s and ℓ that are both integers and at least 1 which yield $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$. $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) = -\frac{1}{2}k_\circ \ell^2 + \ell(s\bar{\tau} - \frac{1}{2}k_\circ) + s\bar{\tau} - \frac{1}{2}s(s+1)n$ is a second-degree polynomial in ℓ with negative leading term coefficient, so for a given s the real maximal is found by differentiating and setting to zero:

$$\ell_{\mathrm{top}}(s) = \frac{s\bar{\tau}}{k_\circ} - \frac{1}{2}$$

This is not usually integer. Due to the parabolic shape of $E_{\mathrm{GS}}^{[n,k]}$ as a function of ℓ, the integer value of ℓ which maximises $E_{\mathrm{GS}}^{[n,k]}$, $\ell_{\mathrm{int}}(s)$, is then at most $\frac{1}{2}$ from ℓ_{top}, namely $\ell_{\mathrm{int}}(s) = \lceil \ell_{\mathrm{top}}(s) - \frac{1}{2} \rceil$. We'll get back to whether $\ell_{\mathrm{int}}(s) \geq 1$. Also, any value of ℓ farther from $\ell_{\mathrm{top}}(s)$ results in a lower value of $E_{\mathrm{GS}}^{[n,k]}$. Therefore, if we find an s such that $E_{\mathrm{GS}}^{[n,k]}(s, \tilde{\ell}(s), \tau) > 0$ for $\tilde{\ell}(s) = \frac{s\bar{\tau}}{k_\circ}$, then we can be sure that $E_{\mathrm{GS}}^{[n,k]}(s, \ell_{\mathrm{int}}(s), \tau) > 0$. Now we have $E_{\mathrm{GS}}^{[n,k]}(s, \tilde{\ell}(s), \tau) > 0$ exactly when

$$s > \frac{\tau k_\circ}{\bar{\tau}^2 - n k_\circ} = s_{\min}$$

whenever $\bar{\tau}^2 > n k_\circ$ which it is exactly when $\tau < n - \sqrt{n(n-d)}$. Therefore, choosing s as $s_{\mathrm{int}} = \lfloor s_{\min} + 1 \rfloor \geq 1$, we are guaranteed that choosing ℓ as $\ell_{\mathrm{int}}(s_{\mathrm{int}})$, we will satisfy $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$. Since $\tau \leq d = n - (k-1)$ we have $\bar{\tau} = n - \tau \geq k - 1 = k_\circ$ and so $\ell_{\mathrm{top}}(s_{\mathrm{int}}) \geq s_{\mathrm{int}} - \frac{1}{2} \geq \frac{1}{2}$ which gives $\ell_{\mathrm{int}}(s_{\mathrm{int}}) \geq 1$.

We can find a better list size, though. For $s = s_{\mathrm{int}}$, we now know there exists at least one positive integer value of ℓ, namely $\ell_{\mathrm{int}}(s_{\mathrm{int}})$, such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$. We would like to select the smallest, so we solve $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ for ℓ and get

$$\ell \in \left] \tfrac{\bar{\tau}}{k_\circ} s - \tfrac{1}{2} - \tfrac{\sqrt{\tilde{D}}}{2k_\circ} \;,\; \tfrac{\bar{\tau}}{k_\circ} s - \tfrac{1}{2} + \tfrac{\sqrt{\tilde{D}}}{2k_\circ} \right[$$
$$\tilde{D} = (2s\bar{\tau} - k_\circ)^2 + 4k_\circ s(2\bar{\tau} - ns - n)$$

For $s = s_{\mathrm{int}}$, we know that $\ell_{\mathrm{int}}(s_{\mathrm{int}})$ must be in the above range for ℓ, so it is both non-empty and contains at least one positive integer. By the original expression for $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau)$, it is clear that the above range cannot include $\ell = 0$ whenever $s > 0$, which means that the entire range must be positive numbers. Therefore, we can choose ℓ as the smallest integer in the range. After some rewriting, this expression becomes that of the proposition. □

Remark. Obviously, this is not the first time closed expressions for the parameters s and ℓ for the Guruswami–Sudan algorithm have been found, but surprisingly—at least to me—I have not been able to find any *good* and *correct* closed expressions! In the original article [GS99], a quick analysis yielding expressions for s and ℓ that grow asymptotically in an acceptable manner was given, but these are obviously unsuitable for use or non-asymptotic analysis. McEliece made a lengthy analysis of the algorithm and its parameters [McE03], but the expressions are not always correct! (see below). I have not been able to find any other analyses of concrete parameter choices.

Of course, for given parameters and a given τ, one can simply brute force search for the smallest ℓ such that an s exists which give $E(s,\ell,\tau) > 0$. On Figure 3.2, we have depicted a comparison of all these list size values for a quite long code; this also demonstrates that fallibility of McEliece's expressions. ♦

Remark. As mentioned above, one can brute force search for the smallest ℓ such an s exist with $E(s,\ell,\tau) > 0$, and one can of course also search for a smallest s such that an ℓ exists with $E(s,\ell,\tau) > 0$. I have found no examples where these two strategies do *not* yield the same, so I conjecture that they always do.

Going to real analysis, the points in the (s,ℓ) plane given by $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$ are "inside" the branches of a hyperbola, with one branch completely in $\ell, s > 0$ and the other outside. The asymptotes of the hyperbola could be shown to both have positive, finite slope and are quite close to each other, making the branches relatively sharp around the centers. I believe my conjecture is paramount to showing that whenever the region within the "positive" branch contains two $\mathbb{Z}^2$ lattice points $(s+1,\ell)$ and $(s,\ell+1)$, it also contains (s,ℓ). This should be true if the asymptotes of the branches are as I described them, in precise terms. ♦

Example 3.12. *For the* $[250, 70, 181]$ *code from Example 3.5 on page 50, we mentioned parameter choices for various decoding radii; these are all minimally possible (determined by brute force search). Proposition 3.11 also gives these for* $\tau = 105$ *and* $\tau = 118$*, while for* $\tau = 97$ *it yields* $(s,\ell) = (2,3)$. ♠

For the sake of the upcoming asymptotic analysis, we can obtain a nice upper bound on the size of ℓ:

Corollary 3.13. *The choice of* s *and* ℓ *given by Proposition 3.11 satisfies*

$$\ell \leq \tfrac{\bar{\tau}}{k_\circ} s_{\min} + 1 = \frac{\tau\bar{\tau}}{\bar{\tau}^2 - nk_\circ} + 1$$

Proof. Let $\varepsilon = s - s_{\min}$. Then we have for the chosen value of ℓ:

$$\ell \leq \tfrac{\bar{\tau}}{k_\circ} s_{\min} + \tfrac{1}{2} + \tfrac{\bar{\tau}}{k_\circ}\varepsilon - \tfrac{1}{k_\circ}\sqrt{\varepsilon^2(\bar{\tau}^2 - nk_\circ) + \varepsilon k_\circ \tau + \tfrac{k_\circ^2}{4}} \tag{3.2}$$

The last two terms turn out to be at most a small positive number. In particular

$$\tfrac{\bar{\tau}}{k_\circ}\varepsilon - \tfrac{1}{k_\circ}\sqrt{\varepsilon^2(\bar{\tau}^2 - nk_\circ) + \varepsilon k_\circ \tau + \tfrac{k_\circ^2}{4}} \leq \tfrac{1}{2} \quad \Longleftrightarrow$$
$$\varepsilon n(1-\varepsilon) \geq 0$$

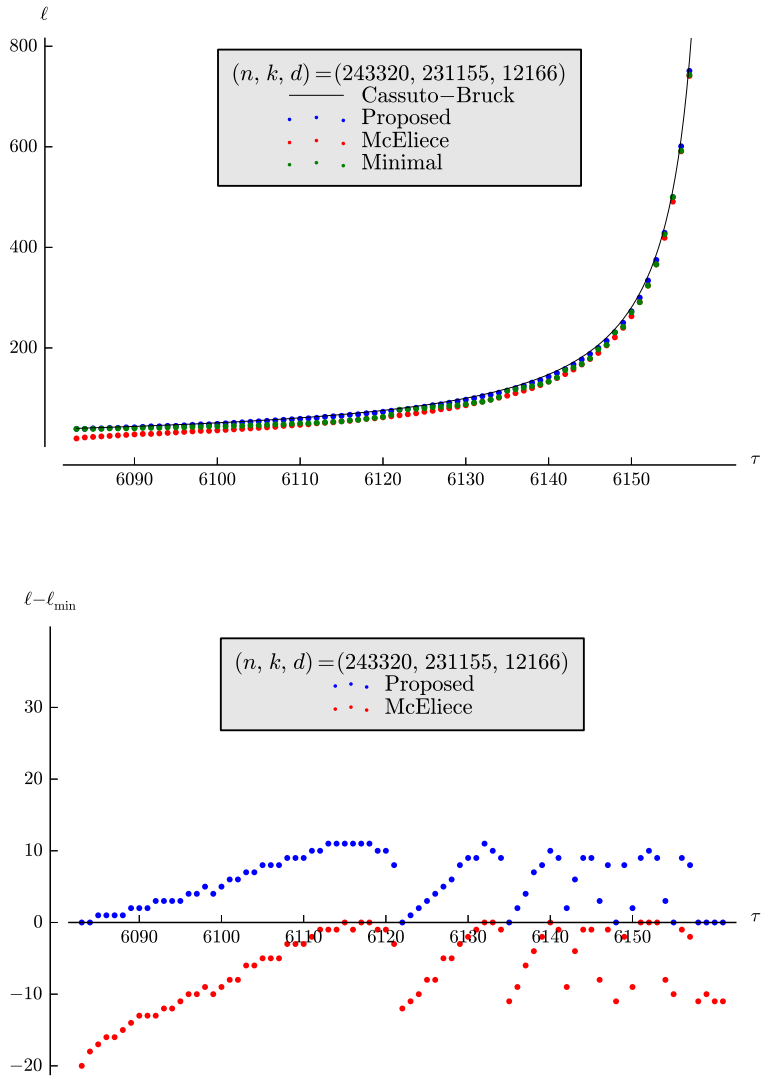

Figure 3.2: Comparison between the list sizes computed by Proposition 3.11 and by McEliece's expressions [McE03] for code parameters chosen for their suitable list-decoding range. "Minimal" is the minimal ℓ for the given τ such that $\exists s \,.\, E_{\mathrm{GS}}^{[n,k]}(s, \ell, \tau) > 0$, obtained by brute force search. Above we see absolute list sizes, while below we see the difference to "Minimal".

Note how the sub-optimality of the list size computed by Proposition 3.11 is relatively small compared to the code parameters, and that it stays small relative to the minimal values possible.

Note also that McEliece's expressions quite often gives a list size *below* the minimal possible, demonstrating an error somewhere in the derivations of those expressions.

We have left out the parameters given by Guruswami and Sudan [GS99] since they already for $\tau = 6083$ (the minimal) give $\ell = 1561$, which is way outside the plot area. We have also included the theoretical list size bound by Cassuto and Bruck [CB04].

which is always true, as $0 < \varepsilon \leq 1$. Inserting the inequality back in (3.2) and expanding $s_{\min}$, we get the sought upper bound for ℓ. □

Corollary 3.14. *Let $T = \frac{\tau}{n}$ and $D = \frac{d}{n}$ as well as $J = 1 - \sqrt{1-D}$. Then Proposition 3.11's choices of s and ℓ satisfies*

$$s \leq \frac{(D-T)(1-T)}{(J-T)(2-(J+T))} < \frac{1-T}{1-\frac{T}{J}}$$
$$\ell \leq \frac{(D-T)}{(J-T)(2-(J+T))} < \frac{1}{1-\frac{T}{J}}$$

Proof. Note first that $(J-T)(2-(J+T)) = T^2 - 2T + D$. First for s; we simply relax the flooring function and rearrange:

$$s \leq \frac{\tau k_\circ}{(n-\tau)^2 - n(k-1)} + 1 = \frac{T(1-D)}{(1-T)^2 - (1-D)} + 1 = \frac{T^2 - TD - T + D}{T^2 - 2T + D}$$

which simplifies to the tighter of the bounds. For ℓ we do the same but with the relaxation of Corollary 3.13:

$$\ell \leq \frac{\tau(n-\tau)}{(n-\tau)^2 - n(k-1)} + 1 = \frac{T(1-T)}{(1-T)^2 - (1-D)} + 1 = \frac{D-T}{T^2 - 2T + D}$$

The relaxations of these bounds are found by upper bounding the function $\zeta(T) = \frac{D-T}{2-(J+T)}$, defined on the range $0 < T < J$. The derivative of ζ with regards to T is always negative, so the maximal value of ζ must be in the limit for $T \to 0$, yielding

$$\lim_{T\to 0} \zeta(T) = \frac{D}{2-J} = \frac{J(2-J)}{2-J} = J$$

Replacing the term $\frac{D-T}{2-(J+T)}$ in the above expressions for ℓ and s with J yields the corollary's final inequalities. Note that this approximation is good for low values of T, but gets increasingly bad for $T \to J$, where $\zeta(T) \to \frac{J}{2}$. □

Remark. For $T \to J$, the fraction ℓ/s between our upper bounds is $(1-T)^{-1} \to \sqrt{\frac{n}{k-1}}$, as we had proven would be the limit for ℓ/s in Corollary 3.9. ◆

Remark. In a technical report Cassuto and Bruck [CB04] presented a new upper bound for the number of codewords within a Hamming sphere of some radius $\tau < n - \sqrt{n(n-d)}$ for any linear code with length n and minimum distance d. That bound is exactly the tighter of the bounds for ℓ given in Corollary 3.14! On Figure 3.2 we have also exemplified this closeness. In [CB04] they compare their found bound with the expressions of McEliece [McE03], and show that the difference is "small", but as we have demonstrated, those expressions are not always correct, so that has little value. ◆

For even more overview, a rough asymptotic estimate is sometimes very enlightening:

Corollary 3.15. *Whenever $\tau < n - \sqrt{n(n-d)}$, there exist $s, \ell \in O(n^2)$ such that $E_{\mathrm{GS}}^{[n,k]}(s, \ell, \tau) > 0$.*

Proof. As $k_\circ$ is an integer and $\bar{\tau}^2 > nk_\circ$, we of course have $\bar{\tau}^2 - nk_\circ \geq 1$, and we also have $\tau, \bar{\tau} \in O(n)$; thus by Corollary 3.13, $\ell \in O(n^2)$ which implies $s \in O(n^2)$. □

It should be noted that this is a worst case, occurring when $\bar{\tau}^2 - nk_\circ$ is very small, i.e. τ is very close to the decoding bound. Due to integer rounding, cases like these are not easily found analytically, but they do crop up from time to time.

Example 3.16. *An* $[2480,\ 1489,\ 992]$ *GRS code can be decoded up to* $\tau = 559$*, which would yield* $\bar{\tau}^2 - nk_\circ = 1$*; i.e. the minimum. Using Proposition 3.11, we get the astronomical* $\ell = 1\,073\,840$ *and* $s = 831\,793$*, which are also the minimum parameters for these* n, k, τ*.*

However, for those n *and* k*, decoding just one error less, that is* $\tau = 558$*, we get the much more reasonable* $\ell = 280$ *and* $s = 217$*.* ♠

The above example demonstrates the general behaviour, captured by the following easy corollary:

Corollary 3.17. *Whenever* $\tau \leq n - \sqrt{n(n-d)} - \varepsilon n$ *for some* $0 < \varepsilon < 1$*, there exist* $s, \ell \in O(\varepsilon^{-1})$ *such that* $E_{\mathrm{GS}}^{[n,k]}(s, \ell, \tau) > 0$*. In particular, if* $\varepsilon n = 1$*, then* $\ell, s \in O(n)$*.*

Proof. We get $J - T \geq (1 - \sqrt{1-D}) - (1 - \sqrt{1-D} - \varepsilon) = \varepsilon$. Inserting this in the tighter of the parameter bounds of Corollary 3.14, we get the sought as the remaining terms are in $O(1)$. □

3.1.3 Sudan: the case of $s = 1$

The Guruswami–Sudan algorithm was preceded chronologically by the simpler Sudan algorithm [Sud97], which is exactly the same algorithm but having $s = 1$ always. Since we will be discussing Power decoding in Chapter 4, which has strong ties to this special case of the Guruswami–Sudan, we briefly give some analysis of this case by itself; fortunately, this is much easier than for the general case.

Proposition 3.18. $E_{\mathrm{GS}}^{[n,k]}(1, \ell, \tau) > 0$ *whenever* $\tau < f(\ell)$ *where* $f(\ell) = \frac{\ell}{\ell+1} n - \frac{1}{2}\ell(k-1)$*. Regarding* $f : \mathbb{N} \mapsto \mathbb{R}$ *then it satisfies* $f(\ell) < n + \frac{1}{2}(k-1) - \sqrt{2n(k-1)}$ *and is maximal either when* $\ell = \left\lfloor \sqrt{\frac{2n}{k-1}} - 1 \right\rfloor$ *or* $\ell = \left\lceil \sqrt{\frac{2n}{k-1}} - 1 \right\rceil$*. In particular, if* $\frac{k-1}{n} \geq \frac{1}{3}$ *then* $f(\ell)$ *is maximal at* $\ell = 1$*.*

Proof. We have $E_{\mathrm{GS}}^{[n,k]}(1, \ell, \tau) = (\ell+1)(n-\tau) - n - \frac{1}{2}\ell(\ell+1)(k-1)$, and requiring this to be greater than zero immediately leads to $\tau < f(\ell)$. Going to real analysis then since $f'(\ell) = \frac{n}{(\ell+1)^2} - \frac{1}{2}(k-1)$ this means that the only stationary point for positive, real ℓ is $\ell = \sqrt{\frac{2n}{k-1}} - 1$. It is easy to see that this is a maximal point, from where the statement on maximal integral ℓ immediately follows. The maximal value of $f(\ell)$ is then $f(\sqrt{\frac{2n}{k-1}} - 1)$ which simplifies to $n + \frac{1}{2}(k-1) - \sqrt{2n(k-1)}$.

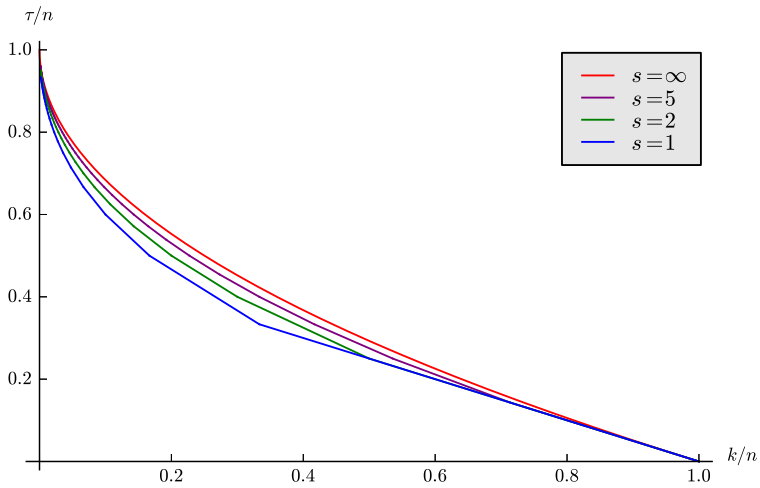

Figure 3.3: Decoding radii for various multiplicities. Notice how the graphs are piece-wise linear. For $s = 1$, i.e. the Sudan case, we also notice that the first piece to lift from the half-minimum distance line begins at $k/n = 1/3$, as stated by Proposition 3.18.

For the last statement, notice simply that if $f(\ell)$ is *not* maximal at $\ell = 1$ then $f(2) > f(1) = \frac{n-k+1}{2}$ which is the same as $\frac{2}{3}n - (k-1) > \frac{n-k+1}{2}$, i.e. $\frac{k-1}{n} < \frac{1}{3}$. □

See Figure 3.3 for a depiction of the decoding radius compared to higher multiplicities and the Johnson bound.

Example 3.19. *For the [250, 70, 181] code of Example 3.5 on page 50, we have* $\sqrt{\frac{2n}{k-1}} \approx 1.69$ *and* $f(1) = 90\frac{1}{2}$ *while* $f(2) = 97\frac{2}{3}$. *So, as we already knew,* $(s, \ell) = (1, 2)$ *allows us to decode* 97 *errors.* ♠

Example 3.20. *We will meet the Sudan case again, in particular for Power decoding in Chapter 4, so we will introduce a lower rate code for better exemplification: so consider instead a [250, 40, 211] GRS code. Half-the-minimum distance decoding would be* 105 *errors. We have* $\sqrt{\frac{2n}{k-1}} \approx 2.58$ *and* $f(2) = 127\frac{2}{3}$ *while* $f(3) = 129$*; thus* $(s, \ell) = (1, 3)$ *allows us to decode* 129 *errors. Going to higher multiplicities, we could have the quite steep* $(s, \ell) = (116, 293)$ *decoding* 151 *errors.* ♠

3.2 Finding Q in an explicit module

We will now show an alternative manner in which to find a $Q \in \mathbb{F}[x, y]$ satisfying Theorem 3.2 on page 48 for a given decoding instance. The method will turn out to be much faster than the Gaussian elimination approach, and will of course draw upon the module minimisation techniques we developed in Chapter 2. We will need

to assume $s \leq \ell$ which by Lemma 3.7 on page 51 is no real restriction.

Definition 3.21. Let $\mathcal{M}_{s,\ell} \subset \mathbb{F}[x,y]$ denote the space of all bivariate polynomials passing through the points $(\alpha_1, r'_1), \ldots, (\alpha_n, r'_n)$ with multiplicity s and with y-degree at most ℓ, i.e. satisfying Point 1 of Theorem 3.2.

Finding a $Q \in \mathbb{F}[x,y]$ for satisfying the requirements of Theorem 3.2 is then the same as finding an element in $\mathcal{M}_{s,\ell}$ with low enough $(1, k-1)$-weighted degree. Since we know that an element with *minimal* $(1, k-1)$-weighted degree will at least be low enough, we will find one such.

Following the ideas of Lee and O'Sullivan [LO08], we can first remark that $\mathcal{M}_{s,\ell}$ is an $\mathbb{F}[x]$ module. Second, we can give an explicit basis for $\mathcal{M}_{s,\ell}$. First we give two definitions which will be used repeatedly in the remainder of the thesis:

Definition 3.22. For a given GRS code, define $G(x) = \prod_{i=1}^{n}(x - \alpha_i)$. For a given received word for this code, define $R(x)$ in $\mathbb{F}[x]$ as the unique Lagrange interpolation polynomial going through the points (α_i, r'_i) for $i = 1, \ldots, n$. Thus

$$R(x) = \sum_{i=1}^{n} r'_i \prod_{j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j}$$

Denote by $Q_{[t]}(x)$ the y^t-coefficient of $Q(x,y)$ when Q is regarded over $\mathbb{F}[x][y]$.

Lemma 3.23. *Let $Q \in \mathcal{M}_{s,\ell}$ with $\deg_y Q = t < s$. Then $G(x)^{s-t} \mid Q_{[t]}(x)$.*

Proof. $Q(x,y)$ interpolates the n points (α_i, r'_i) with multiplicity s, so for any i, $Q(x+\alpha_i, y+r'_i) = \sum_{j=0}^{t} Q_{[j]}(x+\alpha_i)(y+r'_i)^j$ has no monomials of total degree less than s. Multiplying out the $(y+r'_i)^j$-terms, $Q_{[t]}(x+\alpha_i)y^t$ will be the only term with y-degree t. Therefore $Q_{[t]}(x+\alpha_i)$ can have no monomials of degree less than $s-t$, which implies $(x-\alpha_i)^{s-t} \mid Q_{[t]}(x)$. As this holds for any i, we proved the lemma. □

Theorem 3.24. *The module $\mathcal{M}_{s,\ell}$ is generated as an $\mathbb{F}[x]$-module by the $\ell+1$ polynomials $P^{(i)} \in \mathbb{F}[x,y]$ given by*

$$P^{(t)} = G^{s-t}(y-R)^t, \qquad \text{for } 0 \leq t \leq s,$$
$$P^{(t)} = y^{t-s}(y-R)^s, \qquad \text{for } s < t \leq \ell.$$

Proof. It is easy to see that each $P^{(t)} \in \mathcal{M}_{s,\ell}$ since both G and $y-R$ go through the n points (α_i, r'_i) with multiplicity one, and that G and $y-R$ divide $P^{(t)}$ with total power s for each t.

To see that any element of $\mathcal{M}_{s,\ell}$ can be written as an $\mathbb{F}[x]$-combination of the $P^{(t)}$, let $Q(x,y)$ be some element of $\mathcal{M}_{s,\ell}$. Then the polynomial $Q^{(\ell-1)}(x,y) = Q - Q_{[\ell]}(x)P^{(\ell)}$ has y-degree at most $\ell-1$. Since both Q and $P^{(\ell)}$ are in $\mathcal{M}_{s,\ell}$, so must $Q^{(\ell-1)}$ be in $\mathcal{M}_{s,\ell}$. Since $P^{(t)}$ has y-degree t and $P^{(t)}_{[t]}(x) = 1$ for $t = \ell, \ell-1, \ldots, s$, we can continue reducing this way until we reach a $Q^{(s-1)}(x,y) \in \mathcal{M}_{s,\ell}$ with y-degree at

most $s-1$. From then on, we have $P^{(t)}_{[t]}(x) = G^{s-t}$, but by Lemma 3.23, we must also have $G(x) \mid Q^{(s-1)}_{[s-1]}(x)$, so we can also reduce by $P^{(s-1)}$. This can be continued with the remaining $P^{(t)}$, eventually reducing the remainder to 0. □

Instead of viewing $\mathcal{M}_{s,\ell}$ as an $\mathbb{F}[x]$ module of bivariate polynomials with y-degree at most $\ell+1$, we can view it as a module of $\mathbb{F}[x]$-vectors of length $\ell+1$, so that the ith position of an element corresponds to the $\mathbb{F}[x]$-coefficient to y^{i-1}; obviously, this retains all $\mathbb{F}[x]$ operations. Similarly, any basis of $\mathcal{M}_{s,\ell}$ can be represented as a matrix whose rows are such representations of the basis elements. More specifically, the basis of Theorem 3.24 becomes the following matrix:

$$A_{s,\ell} = \begin{pmatrix} G^s & & & & & & \\ G^{s-1}(-R) & G^{s-1} & & & & 0 & \\ \vdots & & \ddots & & & & \\ (-R)^s & \binom{s}{1}(-R)^{s-1} & \dots & 1 & & & \\ & (-R)^s & & \dots & 1 & & \\ & & \ddots & & & \ddots & \\ 0 & & & (-R)^s & \dots & \binom{s}{s-1}(-R) & 1 \end{pmatrix} \tag{3.3}$$

Note that for any $Q = \sum_{t=0}^{\ell} Q_t(x)y^t$ then $\deg_{1,k-1} Q = \deg \Phi_{1,\boldsymbol{w}}\big((Q_0,\dots,Q_\ell)\big)$, where $\boldsymbol{w} = \big(0, k-1, \dots, \ell(k-1)\big)$ and $\Phi_{\nu,\boldsymbol{w}}$ are from Definition 2.11 on page 16. We are thus looking for a vector in the module spanned by the rows of $A_{s,\ell}$ which is minimal under the module monomial ordering $\preceq_{1,\boldsymbol{w}}$. By Corollary 2.15 on page 18, if $B_{s,\ell}$ is a basis of $\mathcal{M}_{s,\ell}$ in $\Phi_{1,\boldsymbol{w}}(B_{s,\ell})$-weighted weak Popov form, then the row of $B_{s,\ell}$ which orders minimal according to $\preceq_{1,\boldsymbol{w}}$ must be such a vector. Thus, we can solve the problem by applying any of the algorithms in Chapter 2 which can bring $\Phi_{1,\boldsymbol{w}}(A_{s,\ell})$ to weak Popov form.

For complexity estimates, let us investigate some key properties of $A_{s,\ell}$:

Lemma 3.25. *Let* $\boldsymbol{w} = \big(0, k-1, \dots, \ell(k-1)\big)$. *Then*

$$\begin{aligned} \operatorname{maxdeg}\left(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})\right) &= sn \\ \Delta(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})) &= \tfrac{1}{2}(2\ell - s + 1)s(\deg R - k + 1) \;<\; \ell s(n-k) \end{aligned}$$

Proof. Since $\deg R \le n-1$ and $k-1 < n$, the max-degree of $\Phi_{1,\boldsymbol{w}}(A_{s,\ell})$ follows immediately as the degree of the upper left-hand element. For the orthogonality defect, let us compute both the degree and the determinant.

For the former, we have $\deg(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})) = \sum_{i=0}^{\ell} \deg_{1,k-1} P^{(t)}(x,y)$, where the $P^{(t)}$ are as in Theorem 3.24. Recall the definition of the positive function $\operatorname{pos}(\cdot)$, see e.g. Appendix A; we can then write $P^{(t)} = G^{\operatorname{pos}(s-b)}(x)(y - R(x))^{t-\operatorname{pos}(t-s)} y^{\operatorname{pos}(t-s)}$. Note that whenever $\mathcal{E} \neq \emptyset$, then $\deg R \ge k$; for otherwise $\boldsymbol{r} = \operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(R) \in \mathcal{C}$. Therefore, $\deg_{1,k-1}(y-R)^t = t \deg R$ and so

$$\deg_{1,k-1} P^{(t)} = \operatorname{pos}(s-t)n + \operatorname{pos}(t-s)(k-1) + \min(t,s)\deg R \tag{3.4}$$

This gives

$$\deg(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})) = \tbinom{s+1}{2}n + \tbinom{\ell-s+1}{2}(k-1) + \Big(\tbinom{s+1}{2} + (\ell-s)s\Big)\deg R$$

Since $A_{s,\ell}$ is lower triangular, the determinant is easy:

$$\det(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})) = \prod_{t=0}^{\ell} G^{\mathrm{pos}(s-t)}x^{t(k-1)} \qquad \text{and so}$$
$$\deg\det(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})) = \tbinom{s+1}{2}n + \tbinom{\ell+1}{2}(k-1)$$

The orthogonality defect can then be simplified to

$$\begin{aligned}\Delta(\Phi_{1,\boldsymbol{w}}(A_{s,\ell})) &= \tbinom{\ell-s+1}{2}(k-1) + \Big(\tbinom{s+1}{2} + (\ell-s)s\Big)\deg R - \tbinom{\ell+1}{2}(k-1)\\ &= \deg R\big(s\ell - \tfrac{1}{2}s^2 + \tfrac{1}{2}s\big) - \tfrac{1}{2}(k-1)\big(\ell^2 + \ell - (\ell-s+1)(\ell-s)\big)\\ &= \tfrac{1}{2}(2\ell - s + 1)s(\deg R - k + 1)\end{aligned}$$ □

Lemma 3.26. *Computing $A_{s,\ell}$ can be done in $O\big(s^2\ell n + s^2\mathrm{P}(sn) + \mathrm{P}(n)\log n\big)$.*

Proof. Computing R and G by Lagrangian interpolation can be done in complexity $O(\mathrm{P}(n)\log n)$, see e.g. [vzGG03, p. 297]. We can then compute all expressions R^iG^j for $0 \le i+j \le s$ iteratively in $O(s^2\mathrm{P}(sn))$. Each of the non-zero entries of $A_{s,\ell}$ is a constant multiple of such an expression and can thus be computed in $O(sn)$ afterwards; the total cost follows by remarking that there are fewer than $(s+1)(\ell+1)$ non-zero entries. □

Proposition 3.27. *The worst-case complexity of finding a satisfactory interpolation polynomial Q by minimising $\Phi_{1,\boldsymbol{w}}(A_{s,\ell})$ is as in Table 3.1, for various choices of module minimisation algorithm.*

Proof. This follows from Table 2.3 on page 42 since we need only one row of the basis in $\Phi_{1,\boldsymbol{w}}$-weighted weak Popov form, and using Lemma 3.25. It should be noted that the cost of constructing $A_{s,\ell}$ is dominated by the cost of minimising it, no matter which method we use for the latter. □

Remark. We see that focusing on n, then the GJV is fastest with approximately $O\big(\ell^3 sn\log(n)^{O(1)}\big)$. This is the fastest known interpolation method, and it has been described in roughly equivalent ways before, see Section 3.6. One can note, however, that if $n-k \ll n$ then the Alekhnovich algorithm looks more favourable since most of the work will be done on truncated polynomials. ♦

Example 3.28. *Consider the $[250, 70, 181]$ code from Example 3.5 on page 50 decoded up to $\tau = 105$ choosing $(s,\ell) = (2,4)$. We would set $\boldsymbol{w} = (0, 69, 138, 207, 276)$. Assuming a received word yielding the usual $\deg R = n-1$, the matrix $A_{2,4}$ satisfies the following:*

$$A_{2,4} \trianglelefteq \begin{pmatrix} 500 & \bot & \bot & \bot & \bot \\ 499 & 250 & \bot & \bot & \bot \\ 498 & 249 & 0 & \bot & \bot \\ \bot & 498 & 249 & 0 & \bot \\ \bot & \bot & 498 & 249 & 0 \end{pmatrix} \qquad \begin{aligned}\mathrm{maxdeg}\,(\Phi_{1,\boldsymbol{w}}(A_{2,4})) &= 500\\ \Delta(\Phi_{1,\boldsymbol{w}}(A_{2,4})) &= 1260\end{aligned}$$ ♠

Complexity of computing Q for Guruswami–Sudan by minimising $A_{s,\ell}$

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^3 s^2 n(n-k)$	$\ell^3 s^2 n^2$
Alekhnovich	$\mathrm{M}(\ell)\mathrm{P}(\ell s(n-k))\log(\ell s(n-k)) + \ell^2\mathrm{P}(sn)$	$\ell^4 sn\log(n)^{2+o(1)}$
GJV	$\mathrm{M}(\ell)\mathrm{P}(sn)\log(\ell sn)^{O(1)}$	$\ell^3 sn\log(n)^{O(1)}$

Table 3.1: In the Relaxed column, we have used $(n-k) \in O(n)$ as well as $s, \ell \in O(n^{O(1)})$, the latter following from Corollary 3.15.

Remark. The "re-encoding transformation" is a technique invented by Kötter and Vardy [KV03b] for reducing the complexity of the Guruswami–Sudan algorithm by, reputedly, "replacing n with $n-k$ in the complexity estimate": one adds a codeword to the received word before decoding, to force at least k positions to zero; decoding this modified received word obviously leads to a decoding of the original. Assuming that one chose the first k position, the result is that $g(x) = \prod_{i=0}^{k-1}(x-\alpha_i)$ divides both R and G, where R and G are as in Definition 3.22 but for the modified received word. Regarding $A_{s,\ell}$, we see that we could then divide out g^i in the first i columns, and the row space of the resulting matrix $\grave{A}_{s,\ell}$ would be isomorphic to that of $A_{s,\ell}$. Thus, finding a minimal weighted-degree vector in the row-space of $\grave{A}_{s,\ell}$ would also be a minimal weighted-degree vector in the row-space of $A_{s,\ell}$, assuming the weights of the former were corrected appropriately. However, easy calculation shows that exactly this correction of weights leads to $\Delta(\Phi_{1,\grave{\boldsymbol{w}}}(\grave{A}_{s,\ell})) = \Delta(\Phi_{1,\boldsymbol{w}}(A_{s,\ell}))$. Since the last $\ell - s$ rows of $\grave{A}_{s,\ell}$ equal those of $A_{s,\ell}$, also the max-degree is unchanged; thus the running times of all the module minimisation algorithms are essentially unchanged, at least from an asymptotic perspective. When using the Alekhnovich algorithm, this was already remarked by Brander in his thesis [Bra10, p. 68].

Exactly same arise when using the Kötter interpolation method, generalised by Nielsen and Høholdt [NH98], for which the re-encoding transformation was originally designed: the asymptotic complexity estimate is left completely unchanged by the transformation. The superficial complexity discussions in both [KV03b] as well as the much more verbose [KMV11], however, glance over this. Two important points should be made in their defence though: 1) the transformation *does* give a (constant) reduction, one which is more evident when s is close to ℓ (i.e. high rates), and which might very well be quite noticeable for practical parameters; and 2) the method is much more potent when applied to soft-decision decoding, their main area of interest, since one can eliminate to positions of the highest computational cost. ♦

3.3 Step-wise Q-finding and multi-trial decoding

The main computational task in the method of Section 3.2 is to find a basis of $\mathcal{M}_{s,\ell}$ in $\Phi_{1,\boldsymbol{w}}$-weighted weak Popov form, and this can be carried out directly by

the methods of Chapter 2. In this section we will give a step-wise method for doing this, utilising a recurrence between the explicit bases of $\mathcal{M}_{s,\ell}$, i.e. the matrix $A_{s,\ell}$ on page 61, when considering increasing values of s and ℓ. This method also gives the possibility of shaping our list decoder as a maximum-likelihood list decoder.

Since we will be considering series of choices for the parameters s and ℓ for a given code, we introduce the function $\tau(s,\ell)$ as the greatest positive integer such that $E_{\mathrm{GS}}^{[n,k]}\big(s,\ell,\tau(s,\ell)\big) > 0$. For notational convenience, we will also write $\Phi_{1,\circ}$ in place of $\Phi_{1,(1,k-1,\ldots,\hat{\ell}(k-1))}$ where the value of $\hat{\ell}$ is inferred by the objects we apply $\Phi_{1,\circ}$ to.

Using the results of the preceding section, we show in Section 3.3.2 that given a basis $B_{s,\ell}$ of $\mathcal{M}_{s,\ell}$ in $\Phi_{1,\circ}$-weighted weak Popov form, then we can write down a matrix $C^{\mathrm{I}}_{s,\ell+1}$ which is a basis of $M_{s,\ell+1}$ and such that the orthogonality defect of $\Phi_{1,\circ}(\mathcal{C}^{\mathrm{I}}_{s,\ell+1})$ is much lower than that of $\Phi_{1,\circ}(A_{s,\ell+1})$. This means that reducing $\Phi_{1,\circ}(C^{\mathrm{I}}_{s,\ell+1})$ to weak Popov form using the Mulders–Storjohann or Alekhnovich algorithm is faster than reducing $\Phi_{1,\circ}(A_{s,\ell+1})$. We call this kind of refinement a "refinement step of type I". In Section 3.3.3, we similarly give a way to refine a basis of $\mathcal{M}_{s,\ell}$ to one of $\mathcal{M}_{s+1,\ell+1}$, and we call this a "refinement step of type II".

If we first compute a basis in $\Phi_{1,\circ}$-weighted weak Popov form of $\mathcal{M}_{1,1}$ using $A_{1,1}$, we can perform a sequence of refinement steps of type I and II to compute a basis in $\Phi_{1,\circ}$-weighted weak Popov form of $\mathcal{M}_{s,\ell}$, since $s \leq \ell$. After any step, having some intermediate $\hat{s} \leq s$, $\hat{\ell} \leq \ell$, we will thus have a basis of $\mathcal{M}_{\hat{s},\hat{\ell}}$ in $\Phi_{1,\circ}$-weighted weak Popov form, say $B_{\hat{s},\hat{\ell}}$. By Corollary 2.15, we could thus extract from $B_{\hat{s},\hat{\ell}}$ a $\hat{Q}(x,y) \in \mathcal{M}_{\hat{s},\hat{\ell}}$ with minimal $(1,k-1)$-weighted degree. Since $\hat{Q}$ must satisfy the interpolation conditions of Theorem 3.2 for multiplicity $\hat{s}$ and list size $\hat{\ell}$, and since the weighted degree is minimal among such polynomials, it must also satisfy the degree constraints for $\hat{\tau} = \tau(\hat{s},\hat{\ell})$. By that theorem any codeword with distance at most $\hat{\tau}$ from $\boldsymbol{r}$ would then be represented by a root of $\hat{Q}(x,y)$.

Algorithm 5 is a generalisation and formalisation of this method. For a given GRS code, one chooses ultimate parameters (s,ℓ,τ) with $s \leq \ell$ and such that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) > 0$. One also chooses a list of refinement steps and chooses after which steps to attempt decoding; these choices are represented by a list of $\mathsf{S}_1, \mathsf{S}_2$ and Root elements. This list must contain exactly $s-\ell$ S_1-elements and $s-1$ S_2-elements, as it begins by computing a basis for $\mathcal{M}_{1,1}$ and will end with a basis for $\mathcal{M}_{s,\ell}$. Whenever there is a Root element in the list, the algorithm finds all codewords with distance at most $\hat{\tau} = \tau(\hat{s},\hat{\ell})$ from $\boldsymbol{r}$; if this list is non-empty, the computation breaks and the list is returned.

The algorithm calls sub-functions which we explain informally: $\mathsf{RefinementStep2}$ and $\mathsf{RefinementStep2}$ will take $\hat{s},\hat{\ell}$ and a basis $B_{\hat{s},\hat{\ell}}$ of $\mathcal{M}_{\hat{s},\hat{\ell}}$ in $\Phi_{1,\circ}$-weighted weak Popov form, and it then returns a basis for $\mathcal{M}_{\hat{s},\hat{\ell}+1}$ respectively $\mathcal{M}_{\hat{s}+1,\hat{\ell}+1}$ also in $\Phi_{1,\circ}$-weighted weak Popov form; more detailed descriptions for these are given in Section 3.3.2 and Section 3.3.3. $\mathsf{MinimalWeightedRow}$ finds a polynomial of mini-

Algorithm 5 Multi-Trial Guruswami–Sudan Decoding

Input: The received word $\boldsymbol{r} = (r_1, \dots, r_n)$. Ultimate parameters τ, s, ℓ such that $E_{\mathrm{GS}}^{[n,k]}(s, \ell, \tau) > 0$ for the given code. A list C with elements in $\{\mathsf{S}_1, \mathsf{S}_2, \mathsf{Root}\}$ with $\ell - s$ instances of S_1, $s-1$ instances of S_2, and ending with a Root element.

Output: A list of all $f \in \mathbb{F}[x]$ with $\deg f < k$ such that $\mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f) \in \mathcal{C}$ and $\mathrm{dist}(\mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f), \boldsymbol{r})$ is minimal and at most τ. Fail if there are none such.

```
1  Construct A_{1,1} as in (3.3)
2  Module minimise A_{1,1} to obtain a basis in Φ_{1,∘}-weighted weak Popov form, B_{1,1}
3  (ŝ, ℓ̂) ← (1, 1)
4  for each c in C do
5      if c = S_1 then
6          B_{ŝ,ℓ̂+1}  ← RefinementStep1(ŝ, ℓ̂, B_{ŝ,ℓ̂})
7          (ŝ, ℓ̂)     ← (ŝ, ℓ̂ + 1)
8      if c = S_2 then
9          B_{ŝ+1,ℓ̂+1} ← RefinementStep2(ŝ, ℓ̂, B_{ŝ,ℓ̂})
10         (ŝ, ℓ̂)     ← (ŝ + 1, ℓ̂ + 1)
11     if c = Root then
12         Q(x, y)   ← MinimalWeightedRow(B_{ŝ,ℓ̂})
13         if RootFinding(Q, τ(ŝ, ℓ̂)) ≠ ∅ then return this list
14 return Fail
```

mal $(1, k-1)$-weighted degree in $\mathcal{M}_{\hat{s},\hat{\ell}}$ given a basis in $\Phi_{1,\circ}$-weighted weak Popov form (Corollary 2.15). Finally, $\mathsf{RootFinding}(Q, \hat{\tau})$ corresponds to Line 2 and Line 3 of Algorithm 4: it returns all y-roots of $Q(x, y)$ of degree less than k and whose corresponding codeword has distance at most $\hat{\tau}$ from the received word $\boldsymbol{r}$.

3.3.1 The possible refinement paths

There is a large amount of flexibility in the algorithm on how to choose the list C. The two extreme cases are perhaps the most generally interesting: the one without any Root elements except at the end, i.e. usual list decoding; and the one with a Root element each time the decoding radius $\tau(\hat{s}, \hat{\ell})$ has increased, i.e. a variant of maximum-likelihood decoding up to a certain radius.

In Section 3.3.4, we discuss complexity concerns with regards to the chosen path, but it turns out that the price of either type of refinement is very comparable. Therefore, the computations for the refinements in the worst case are not of much significance; however, in the case where we have multiple Root elements, we want to minimise the average computation cost: considering that few errors occur much more frequently than many, we are then interested in reaching each intermediate decoding radius after as few refinements as possible.

Since we do not have a refinement which increases only s, we are inherently limited in the possible paths we can choose. The strongest condition we would like to have satisfied is the following: Let $\lfloor \frac{d-1}{2} \rfloor \leq \tau_1 < \ldots < \tau_m = \tau$ be the series of intermediate decoding radii where one would like to decode. Let (s_i, ℓ_i) be chosen such that $E_{\mathrm{GS}}^{[n,k]}(s_i, \ell_i, \tau_i) > 0$ and either s_i or ℓ_i is minimal possible. Can then the sequence of parameters (s_i, ℓ_i) be reached by refinement steps of type I or II?

I do not have a proof of this statement, but I conjectured with Alexander Zeh in [NZ13] that it is true, and it has been verified for a large number of parameters.

Example 3.29. *Consider the* $[250, 70, 181]$ *code from Example 3.5 on page 50 decoded up to* $\tau = 105$ *choosing* $s, \ell = (2, 4)$*. Not considering* Root *steps, we could choose to apply steps* $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_1$*, yielding the following* (s, ℓ) *refinement path*

$$(1,1) \to (1,2) \to (2,3) \to (2,4)$$

This gives the intermediate decoding radii $90, 97, 104, 105$ *after each step respectively. We could instead choose* $\mathsf{S}_1, \mathsf{S}_1, \mathsf{S}_2$*, yielding the refinement path*

$$(1,1) \to (1,2) \to (1,3) \to (2,4)$$

This results in decoding radii $90, 97, 83, 105$*. This demonstrates that choosing the right refinement path is very important with respect to achieving high decoding radii as early as possible.* ♠

3.3.2 Refinement step type I: $(s, \ell) \mapsto (s, \ell + 1)$

Lemma 3.30. *If* $B^{(0)}, \ldots, B^{(\ell)} \in \mathbb{F}[x, y]$ *is a basis of* $\mathcal{M}$*, then the following is a basis of* $\mathcal{M}_{s,\ell+1}$*:*

$$B^{(0)}, \ \ldots \ , \ B^{(\ell)}, y^{\ell-s+1}(y - R(x))^s.$$

Proof. Investigating the basis of $\mathcal{M}_{s,\ell+1}$ given in Theorem 3.24, we see that the first $\ell + 1$ generators are the generators of $\mathcal{M}_{s,\ell}$. Thus all of these can be described by any basis of $\mathcal{M}_{s,\ell+1}$. The last remaining generator is exactly $y^{\ell-s+1}(y - R(x))^s$. □

In particular, the above lemma holds for a basis of $\mathcal{M}_{s,\ell}$ in $\Phi_{1,\circ}$-weighted weak Popov form. Letting $B_{s,\ell}$ be such a basis, the following matrix thus represents a basis of $\mathcal{M}_{s,\ell+1}$:

$$C_{s,\ell+1}^{\mathrm{I}} = \left(\begin{array}{cccccc|c} & & & B_{s,\ell} & & & \mathbf{0}^T \\ \hline 0 & \ldots & 0 & (-R)^s & \binom{s}{1}(-R)^{s-1} & \ldots & 1 \end{array}\right) \tag{3.5}$$

Lemma 3.31. $\Delta(\Phi_{1,\circ}(C_{s,\ell+1}^{\mathrm{I}})) = s(\deg R - k + 1) \leq s(n - k)$.

Proof. We calculate the two quantities $\det \Phi_{1,\circ}(C^{\mathrm{I}}_{s,\ell+1})$ and $\deg \Phi_{1,\circ}(C^{\mathrm{I}}_{s,\ell+1})$. Since $\Phi_{1,\circ}$ simply scales the ith column with $x^{(i-1)(k-1)}$, it is easy to see that

$$\det \Phi_{1,\circ}(C^{\mathrm{I}}_{s,\ell+1}) = x^{(\ell+1)(k-1)} \det \Phi_{1,\circ}(B_{s,\ell})$$

For the row-degree, it is clearly $\deg \Phi_{1,\circ}(B_{s,\ell})$ plus the row-degree of the last row. If and only if the received word is not a codeword then $\deg R \geq k$, so the leading term of the last row must be $(-R)^s x^{(\ell+1-s)(k-1)}$. Thus, we get

$$\begin{aligned}\Delta(\Phi_{1,\circ}(C^{\mathrm{I}}_{s,\ell+1})) &= \big(\deg \Phi_{1,\circ}(B_{s,\ell}) + s \deg R + (\ell+1-s)(k-1)\big) \\ &\quad - \big(\deg \det \Phi_{1,\circ}(B_{s,\ell}) + (\ell+1)(k-1)\big) \\ &= s(\deg R - k + 1),\end{aligned}$$

where the last step follows from Proposition 2.5 as $\Phi_{1,\circ}(B_{s,\ell})$ is in weak Popov form. □

3.3.3 Refinement step type II: $(s, \ell) \mapsto (s+1, \ell+1)$

Lemma 3.32. *If $B^{(0)}, \ldots, B^{(\ell)} \in \mathbb{F}[x, y]$ is a basis of $\mathcal{M}_{s,\ell}$, then the following is a basis of $\mathcal{M}_{s+1,\ell+1}$:*

$$G^{s+1}(x),\ (y - R(x))B^{(0)},\ \ldots\ ,\ (y - R(x))B^{(\ell)}$$

Proof. Denote by $P^{(0)}_{s,\ell}, \ldots, P^{(\ell)}_{s,\ell}$ the basis of $\mathcal{M}_{s,\ell}$ as given in Theorem 3.24, and by $P^{(0)}_{s+1,\ell+1}, \ldots, P^{(\ell+1)}_{s+1,\ell+1}$ the basis of $\mathcal{M}_{s+1,\ell+1}$. Then observe that for $t > 0$, we have $P^{(t)}_{s+1,\ell+1} = (y - R(x))P^{(t-1)}_{s,\ell}$. Since the $B^{(i)}$ form a basis of $\mathcal{M}_{s,\ell}$, each $P^{(t)}_{s,\ell}$ is expressible as an $\mathbb{F}[x]$-combination of these, and thus for $t > 0$, $P^{(t)}_{s+1,\ell+1}$ is expressible as an $\mathbb{F}[x]$-combination of the $(y - R(x))B^{(i)}$. Remaining is then only $P^{(0)}_{s+1,\ell+1}(x, y) = G^{s+1}(x)$. □

As before, we can use the above with a basis $B_{s,\ell}$ of $\mathcal{M}_{s,\ell}$ in $\Phi_{1,\circ}$-weighted weak Popov form, found in the previous iteration of our algorithm. Remembering that multiplying by y translates to shifting one column to the right in the matrix representation, the following matrix thus represents a basis of $\mathcal{M}_{s+1,\ell+1}$:

$$C^{\mathrm{II}}_{s+1,\ell+1} = \left(\begin{array}{c|c} G^{s+1} & \mathbf{0} \\ \hline \mathbf{0}^T & \mathbf{0} \end{array}\right) + \left(\begin{array}{c|c} 0 & \mathbf{0} \\ \hline \mathbf{0}^T & B_{s,\ell} \end{array}\right) - R \cdot \left(\begin{array}{c|c} \mathbf{0} & 0 \\ \hline B_{s,\ell} & \mathbf{0}^T \end{array}\right). \quad (3.6)$$

Lemma 3.33. $\Delta(\Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1})) = (\ell+1)(\deg R - k + 1) \leq (\ell+1)(n-k)$.

Proof. We compute $\deg \Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1})$ and $\deg \det \Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1})$. For the former, obviously the first row has degree $(s+1)n$. Let $\boldsymbol{b}_i$ denote the ith row of $B_{s,\ell}$. The $(i+1)$th row of $\Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1})$ then has the form

$$\Phi_{1,\circ}\big((0 \mid \boldsymbol{b}_i) - R(\boldsymbol{b}_i \mid 0)\big) = (0 \mid \Phi_{1,\circ}(\boldsymbol{b}_i))x^{k-1} - R(\Phi_{1,\circ}(\boldsymbol{b}_i) \mid 0).$$

As previously mentioned, if and only if the received word is not a codeword, then $\deg R \geq k$. In this case, the leading term of $R\Phi_{1,\circ}(\boldsymbol{b}_i)$ must have greater degree than any term in $x^{k-1}\Phi_{1,\circ}(\boldsymbol{b}_i)$. Thus the degree of the $(i+1)$th row is $\deg R + \deg \Phi_{1,\circ}(\boldsymbol{b}_i)$. Summing up we get

$$\begin{aligned}\deg \Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1}) &= (s+1)n + \sum_{i=0}^{\ell} (\deg R + \deg \Phi_{1,\circ}(\boldsymbol{b}_i)) \\ &= (s+1)n + (\ell+1)\deg R + \deg \Phi_{1,\circ}(B_{s,\ell})\end{aligned}$$

For the determinant, observe that since the top row of $C^{\mathrm{II}}_{s+1,\ell+1}$ has only one non-zero entry, then

$$\begin{aligned}\det \Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1}) &= x^{\binom{\ell+2}{2}(k-1)} \det(C^{\mathrm{II}}_{s+1,\ell+1}) \\ &= x^{\binom{\ell+2}{2}(k-1)} G^{s+1} \det \tilde{B}\end{aligned}$$

where $\tilde{B} = B_{s,\ell} - R\left[\grave{B}_{s,\ell} \mid \mathbf{0}^T\right]$ and $\grave{B}_{s,\ell}$ is all but the zeroth column of $B_{s,\ell}$. This means $\tilde{B}$ can be obtained by starting from $B_{s,\ell}$ and iteratively adding the $(j+1)$th column of $B_{s,\ell}$ scaled by $R(x)$ to the jth column, with j starting from 1 up to ℓ. Since each of these will add a scaled version of an existing column in the matrix, this does not change the determinant. Thus, $\det \tilde{B} = \det B_{s,\ell}$, and this means

$$\begin{aligned}\det \Phi_{1,\circ}(C^{\mathrm{II}}_{s+1,\ell+1}) &= x^{\binom{\ell+2}{2}(k-1)} G^{s+1} \det B_{s,\ell} \\ &= x^{(\ell+1)(k-1)} G^{s+1} \det \Phi_{1,\circ}(B_{s,\ell})\end{aligned}$$

Since $B_{s,\ell}$ is in $\Phi_{1,\circ}$-weighted weak Popov form, we again have $\deg \Phi_{1,\circ}(B_{s,\ell}) = \deg \det \Phi_{1,\circ}(B_{s,\ell})$ and the lemma then follows from the difference of the two calculated quantities. □

Example 3.34. *Choosing the first refinement path in Example 3.29, then in a typical decoding instance $A_{1,1}$ and $B_{1,1}$ would satisfy:*

$$A_{1,1} \trianglelefteq \begin{pmatrix} 250 & \bot \\ 249 & 0 \end{pmatrix} \qquad \Phi_{1,\circ}(B_{1,1}) \trianglelefteq \begin{pmatrix} 160 & 159 \\ 159 & 159 \end{pmatrix} \qquad B_{1,1} \trianglelefteq \begin{pmatrix} 160 & 90 \\ 159 & 90 \end{pmatrix}$$

That yields for the following constructed and minimised matrices:

$$C^{\mathrm{I}}_{1,2} \trianglelefteq \begin{pmatrix} 160 & 90 & \bot \\ 159 & 90 & \bot \\ \bot & 249 & 0 \end{pmatrix} \qquad \Phi_{1,\circ}(B_{1,2}) \trianglelefteq \begin{pmatrix} 153 & 152 & 152 \\ 152 & 152 & 152 \\ 152 & 152 & 151 \end{pmatrix} \qquad B_{1,2} \trianglelefteq \begin{pmatrix} 153 & 83 & 14 \\ 152 & 83 & 14 \\ 152 & 83 & 13 \end{pmatrix}$$

$$C^{\mathrm{II}}_{2,3} \trianglelefteq \begin{pmatrix} 500 & \bot & \bot & \bot \\ 402 & 332 & 263 & 14 \\ 401 & 332 & 263 & 14 \\ 401 & 332 & 262 & 13 \end{pmatrix} \qquad \Phi_{1,\circ}(B_{2,3}) \trianglelefteq \begin{pmatrix} 291 & 291 & 291 & 291 \\ 291 & 291 & 291 & 290 \\ 291 & 291 & 290 & 290 \\ 291 & 290 & 290 & 290 \end{pmatrix} \qquad B_{2,3} \trianglelefteq \begin{pmatrix} 291 & 222 & 153 & 84 \\ 291 & 222 & 153 & 83 \\ 291 & 222 & 152 & 83 \\ 291 & 221 & 152 & 83 \end{pmatrix}$$

$$C^{\mathrm{I}}_{2,4} \trianglelefteq \begin{pmatrix} 291 & 222 & 153 & 84 & \bot \\ 291 & 222 & 153 & 83 & \bot \\ 291 & 222 & 152 & 83 & \bot \\ 291 & 221 & 152 & 83 & \bot \\ \bot & \bot & 498 & 249 & 0 \end{pmatrix} \Phi_{1,\circ}(B_{2,4}) \trianglelefteq \begin{pmatrix} 288 & 288 & 288 & 288 & 288 \\ 288 & 288 & 288 & 288 & 287 \\ 288 & 288 & 288 & 287 & 287 \\ 288 & 288 & 287 & 287 & 287 \\ 288 & 287 & 287 & 287 & 287 \end{pmatrix} B_{2,4} \trianglelefteq \begin{pmatrix} 288 & 219 & 150 & 81 & 12 \\ 288 & 219 & 150 & 81 & 11 \\ 288 & 219 & 150 & 80 & 11 \\ 288 & 219 & 149 & 80 & 11 \\ 288 & 218 & 149 & 80 & 11 \end{pmatrix}$$

Notice that module minimisation "evens out" the degrees, so that e.g. the entries of $\Phi_{1,\circ}(B_{2,4})$ all have similar degrees. The orthogonality defect is related to the "unevenness" of the degrees (of the degrees of the rows, to be more specific); it is therefore interesting to compare the degrees of $C^{\mathrm{I}}_{2,4}$ with those of $A_{2,4}$ on page 62 (remember that the ith column should be weighted by $x^{(i-1)(k-1)}$). ♠

Worst-case complexity of the Multi-trial algorithm

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^3 s^2 n(n-k)$	$\ell^3 s^2 n^2$
Alekhnovich	$\ell \mathrm{M}(\ell)\big(s(n-k)\log^{2+o(1)}(\ell(n-k)) + \mathrm{P}(sn)\big)$	$\ell^4 sn \log(n)^{2+o(1)}$
GJV	$\ell \mathrm{M}(\ell)\mathrm{P}(sn)\log(\ell sn)^{O(1)}$	$\ell^4 sn \log(n)^{O(1)}$

Table 3.2: For relaxation, we have used the same rules as in Table 3.1 on page 63.

3.3.4 Complexity of the method

Using the orthogonality defects obtained for the matrices $C^{\mathrm{I}}_{s,\ell+1}$ and $C^{\mathrm{II}}_{s,\ell+1}$, we can make a precise worst-case asymptotic complexity analysis of Algorithm 5. The average running time will depend on the exact choice of C but we will see that the worst-case complexity will not.

Proposition 3.35. *Given ultimate parameters s, ℓ, then no matter what the choice of C is, the worst-case complexity of Algorithm 5 is as in Table 3.2, for various choices of module minimisation algorithm. In particular, the number of* Root *elements has no influence.*

Proof. The worst-case complexity corresponds to the case that we do not break early but run through the entire list C. Precomputing $A_{1,1}$ can be performed in $O(n \log^2 n \log\log n)$ according Lemma 3.26 on page 62. Reducing to $B_{1,1}$ depends on the module minimisation algorithm, but the relevant measures on $A_{1,1}$ are bounded in Lemma 3.25 on page 61.

Now, C must contain exactly $\ell - s$ S_1-elements and $s-1$ S_2-elements. Using Lemma 3.31 and Lemma 3.33, we can upper bound the complexity of minimisation in each step for any of the minimisation algorithm using Table 2.2 on page 42, by relaxing the intermediate parameters with $\hat{s} \leq s$ and $\hat{\ell} \leq \ell$.

It only remains to count the root-finding steps. Obviously, it never makes sense to have two Root after each other in C, so after removing such possible duplicates, there can be at most ℓ elements Root. When we perform root-finding for intermediate $\hat{s}, \hat{\ell}$, we do so on a polynomial in $\mathcal{M}_{\hat{s},\hat{\ell}}$ of minimal weighted degree, and by the definition of $\mathcal{M}_{\hat{s},\hat{\ell}}$ as well as Theorem 3.2, this weighted degree will be less than $\hat{s}(n-\hat{\tau}) < sn$. Thus we can apply Proposition 3.6 with $N = sn$. □

Note that when using the Mulders–Storjohann or Alekhnovich algorithms for module minimisation, the worst-case complexity of the multi-trial algorithm is the same order as when finding a minimised basis for $\mathcal{M}_{s,\ell}$ immediately, as described in Section 3.2, see Table 3.1. When using the GJV, the low orthogonality defects of $C^{\mathrm{I}}_{\hat{s},\hat{\ell}+1}$ and $C^{\mathrm{II}}_{\hat{s}+1,\hat{\ell}+1}$ provides no immediate benefit, and the total worst-case running time for the multi-trial algorithm is a factor ℓ worse than the one in Section 3.2.

Therefore, considering the Mulders–Storjohann or Alekhnovich algorithm for module

minimisation, we could use any number of intermediate decoding attempts without changing the asymptotic worst-case behaviour. The obvious strategy is then to choose a series of refinement steps such that the decoding radius always increases as early as possible, and attempt decoding after each such increase. The result would be a "maximum-likelihood list decoder", i.e. it finds all *closest* codewords within some ultimate radius $\tau < n - \sqrt{n(n-d)}$. If one is working in a decoding model where such a list suffices, our algorithm will thus have much better average-case complexity since fewer errors occur much more frequently than many, while this will pose no (asymptotic) penalty, compared with using the Mulders–Storjohann or Alekhnovich minimisation algorithms on the final $A_{s,\ell}$ directly. In e.g. software implementations, average-case behaviour is of much higher importance than worst-case behaviour so in this setting our algorithm should be highly useful.

Remark. Consider the case where we use a C with only a Root at the very end, i.e. simply use step-wise interpolation without intermediate decoding attempts, and consider using either the Mulders–Storjohann or Alekhnovich algorithm for module minimisation. Though it is not visible on the worst-case running time, it is ensnaring to believe that the constant for the leading term in the complexity estimates, hidden by the big-O notation, should be lower for the multi-trial algorithm than for the method of Section 3.2: since in the former, the matrices and polynomials gradually increase in size, while in the latter they are of maximal size throughout. It would be interesting to redo the complexity analysis in a non-asymptotic manner, or at least with an asymptotic expansion of at least one term, in order to reveal exactly this most dominant constant. ◆

3.4 Finding Q by key equations

We will now see yet another method to find an interpolation polynomial for the Guruswami–Sudan algorithm: we will rewrite the requirements for a bivariate polynomial to be in $\mathcal{M}_{s,\ell}$ in the form of a 2D key equation, and then we will use the techniques of Section 2.5 to solve it. The rewrite into a 2D key equation is in essence based on the work of Zeh, Gentner, and Augot [ZGA11]; they used a specially tailored extension of the Berlekamp–Massey algorithm to solve the emerging equations.

Though the main idea of the rewrite into a key equation form is straightforward, the details regarding degree constraints and weights for the 2D key equation are unfortunately highly technical. Expressions in n, k, τ, s and ℓ quickly become long and unwieldy; I have opted for introducing a number of short-hand names for the parameters in an attempt to clarify the relation between the various bounds rather than always reducing the expressions to their most primitive form. I hope this will benefit the reader rather than add to the confusion.

As we will see shortly during the derivation, we will need to assume that $x \nmid G(x)$ which is equivalent to 0 not being an evaluation point. This assumption is not

intrinsic to the Guruswami–Sudan algorithm, but it is a common assumption for several other decoders, in particular those based on the traditional key equation for minimum distance decoding; we will use it again in Section 4.2.

For short-hand, introduce $\nabla_t^Q = s(n-\tau) - t(k-1)$. This is the degree constraint on the $\mathbb{F}[x]$-coefficient for y^t in a polynomial which has $(1, k-1)$-weighted degree at most $s(n-\tau)$.

Lemma 3.36. *Consider some $Q = \sum_{t=0}^{\ell} Q_t(x)y^t \in \mathbb{F}[x,y]$ satisfying $\deg_{1,k-1} Q < s(n-\tau)$. Then $Q \in \mathcal{M}_{s,\ell}$ if and only if there exists $B_0, \dots, B_{s-1} \in \mathbb{F}[x]$ such that for $b = 0, \dots, s-1$ then*

$$\sum_{t=b}^{\ell} \tbinom{t}{b} Q_t(x) R^{t-b} = B_b(x) G(x)^{s-b} \tag{3.7}$$

Proof. $Q \in \mathcal{M}_{s,\ell}$ if and only if there exists $B_0, \dots, B_\ell \in \mathbb{F}[x]$ such that $Q = \sum_{t=0}^{\ell} B_t P^{(t)}(x,y)$, where the $P^{(t)}$ are as in Theorem 3.24. Recall as in the proof of Lemma 3.25 on page 61 that $P^{(t)} = G^{\mathrm{pos}(s-b)}(x)(y-R(x))^{t-\mathrm{pos}(t-s)} y^{\mathrm{pos}(t-s)}$. Plugging in $y + R(x)$ in place of y then yields:

$$\begin{aligned}
\sum_{t=0}^{\ell} Q_t(x)(y+R(x))^t &= \sum_{b=0}^{\ell} B_b(x) P^{(b)}(x, y+R(x)) && \iff \\
\sum_{t=0}^{\ell} Q_t(x) \sum_{b=0}^{t} \tbinom{t}{b} y^b R^{t-b}(x) &= \sum_{b=0}^{\ell} B_b(x) G^{\mathrm{pos}(s-b)}(x) y^{b-\mathrm{pos}(b-s)} (y+R(x))^{\mathrm{pos}(b-s)} && \iff \\
\sum_{b=0}^{\ell} y^b \sum_{t=b}^{\ell} \tbinom{t}{b} Q_t(x) R^{t-b}(x) &= \sum_{b=0}^{\ell} B_b(x) G^{\mathrm{pos}(s-b)}(x) y^{b-\mathrm{pos}(b-s)} (y+R(x))^{\mathrm{pos}(b-s)} && (3.8)
\end{aligned}$$

Now, *if* $Q \in \mathcal{M}_{s,\ell}$, then comparing $\mathbb{F}[x]$-coefficients of y^b for low b we get that

$$\sum_{t=b}^{\ell} \tbinom{t}{b} Q_t(x) R^{t-b}(x) = B_b(x) G^{s-b}(x) \qquad\qquad b = 0, \dots, s-1$$

For the other direction, assume that such $B_0, \dots, B_{s-1}$ exist and we need to prove that $Q \in \mathcal{M}_{s,\ell}$, which is paramount to proving that we can find $B_s, \dots, B_\ell$ such that (3.8) holds. But since these new B_b have no influence on the coefficients of y^b for $b = 0, \dots, s-1$, the requirement specialise to only be on the higher degree terms:

$$\sum_{b=s}^{\ell} y^{b-s} \sum_{t=b}^{\ell} \tbinom{t}{b} Q_t(x) R^{t-b}(x) = \sum_{b=s}^{\ell} B_b(x)(y+R(x))^{b-s}$$

Such $B_s, \dots, B_\ell$ can be found since $(y+R(x))^i$ for $i = 0, \dots, \ell - s$ is a basis for $\mathbb{F}[x,y]/\langle y^{\ell-s+1} \rangle$. □

Lemma 3.37. *Let $\nabla_b^B = \nabla_\ell^Q + (\ell - s)n - (\ell - b)$. In the context of Lemma 3.36, then if such $B_0, \dots, B_{s-1}$ exist, they must satisfy $\deg B_b < \nabla_b^B$.*

Proof. Since $\deg Q_t < \nabla_t^Q$, we get by the equation of Lemma 3.36 that

$$\begin{aligned}\deg B_b &\le \max_{t=b}^{\ell}\{\deg Q_t + (t-b)\deg R\} - (s-b)\deg G \\ &< \nabla_\ell^Q + (\ell-b)\deg R - (s-b)n \\ &\le \nabla_\ell^Q + (\ell-b)(n-1) - (s-b)n\end{aligned}$$

since $\deg R > k-1$ whenever errors have occurred. □

Remark. The two preceding lemmas are a paraphrasing of [ZGA11, Proposition 3], where the proof is based on the derivatives of such a $Q(x,y)$, rather than our use of the explicit basis of $\mathcal{M}_{s,\ell}$. The wording of [ZGA11, Proposition 3] contains a small mistake in that the existence of the B_b does not establish an upper bound on the degree of the resulting Q; therefore, it should be part of the assumption of the proposition that $\deg_{1,k-1} Q < s(n-\tau)$. This has no bearing on the rest of their paper. ◆

We will now reach the formulation of finding a satisfactory Q as a solution to a certain 2D key equation; first we prove the algebra, and then in Theorem 3.40 we map the found parameters to the definition of Problem 2.31 on page 26. From Lemma 3.36, such a formulation is quickly found, but we will go on to rewrite this a bit more; the reason is to reduce the size of the resulting key equation, and the result will be basically replacing several powers of ℓ with s in the resulting complexity estimates.

We introduce the operator ${}^{[d_p]}\overline{p}(x)$ for $p \in \mathbb{F}[x]$ with degree at most d_p to mean ${}^{[d_p]}\overline{p}(x) = x^{d_p} p(x^{-1})$. We will often simply write $\overline{p}(x)$ when an implied degree upper bound on p is known. We now introduce a "syndrome" like power series:

$$S_\circ^{(a_R,a_G)}(x) = \frac{\overline{R}(x)^{a_R}}{\overline{G}(x)^{a_G}} \tag{3.9}$$

The exact relation between $S_\circ^{(a_R,a_G)}$ and the usual notion of syndromes, and especially their role in classical decoding algorithms for Reed–Solomon codes, is further explored in Chapter 4.

Proposition 3.38. *Consider a* $Q = \sum_{t=0}^{\ell} Q_t(x) y^t \in \mathbb{F}[x,y]$ *satisfying* $\deg_{1,k-1} Q < s(n-\tau)$. *Then* $Q \in \mathcal{M}_{s,\ell}$ *if and only if there exists* $\tilde{B}_0, \dots, \tilde{B}_{s-1} \in \mathbb{F}[x]$ *with* $\deg \tilde{B}_b < \nabla_b^B - \mu$ *and such that for* $b = 0, \dots, s-1$ *then*

$$\sum_{t=b}^{\ell} \tbinom{t}{b} \overline{Q}_t(x) S^{(t-b,s-b)}(x) x^{u_t+v_t-\mu} \equiv \tilde{B}_b(x) \mod x^{m_b-\mu}$$

where for $b = 0, \ldots, s-1$ *and* $t = 0, \ldots, \ell$*:*

$$\begin{aligned} m_b &= \nabla_\ell^Q + (\ell - b)(n-1) \\ u_t &= (\ell - t)(n-k) \\ v_t &= \operatorname{pos}((t-s)n - (t-1)) \\ \mu &= \min_t\{u_t + v_{t,b}\} \ (= u_s) \\ S^{(t-b,s-b)} &= x^{-v_t}\left((S_\circ^{(t-b,s-b)} \mod x^{m_b-u_t}) - (S_\circ^{(t-b,s-b)} \bmod x^{v_t})\right) \end{aligned}$$

where we for μ *mean that* $\mu = u_s$ *except if* $s \geq k-1$*. Furthermore* $(m_b - \mu) \in O(sn)$*.*

Proof. Note first that either side of (3.7) of Lemma 3.36 has degree less than $m_b = \nabla_\ell^Q + (\ell-b)(n-1)$. (3.7) is true if and only if it is true when coefficients are reversed:

$$\begin{aligned} {}^{[m_b-1]}\overline{\left(\textstyle\sum_{t=b}^{\ell} \binom{t}{b} Q_t(x) R(x)^{t-b}\right)} &= {}^{[m_b-1]}\overline{B_b(x)G(x)^{s-b}} \qquad \Longleftrightarrow \\ \sum_{t=b}^{\ell} \binom{t}{b} \overline{Q}_t(x) \overline{R}(x)^{t-b} x^{u_t} &= \overline{B}_b(x)\overline{G}(x)^{s-b} \end{aligned}$$

where $u_t = (\nabla_\ell^Q + (\ell-b)(n-1) - 1) - (\nabla_t^Q + (t-b)(n-1) - 1) = (\ell - t)(n-k)$. The above is true if and only if it is true when relaxed to a congruence modulo x^{m_b}:

$$\sum_{t=b}^{\ell} \binom{t}{b} \overline{Q}_t(x) \overline{R}(x)^{t-b} x^{u_t} \equiv \overline{B}_b(x)\overline{G}(x)^{s-b} \mod x^{m_b}$$

Since $x \nmid G(x)$, then we can divide by $G(x)^{s-b}$ on both sides of the equation and replace $\overline{R}(x)^{t-b}/\overline{G}(x)^{s-b}$ by $S_\circ^{(t-b,s-b)}(x)$.

Now for the complexity saving reduction. Let $S_\circ^{(t-b,s-b)} = \hat{S}^{(t-b,s-b)} + x^{v_{t,b}} S^{(t-b,s-b)} + O(x^{m_b-u_t})$ with $\deg \hat{S}^{(t-b,s-b)} < v_{t,b}$ and $\deg S^{(t-b,s-b)} < m_b - u_t - v_{t,b}$; we will determine $v_{t,b}$ momentarily. Note that the definition of $S^{(t-b,s-b)}$ corresponds to the proposition's, and note that the big-O part will have no influence on the congruence relation and can be removed. Then we can move lower degrees on the right-hand side to the left-hand side:

$$\begin{aligned} \sum_{t=b}^{\ell} \binom{t}{b} \overline{Q}_t(x) S_\circ^{(t-b,s-b)}(x) x^{u_t} &\equiv \overline{B}_b(x) && \mod x^{m_b} \\ \sum_{t=b}^{\ell} \binom{t}{b} \overline{Q}_t(x) S^{(t-b,s-b)}(x) x^{u_t+v_{t,b}} &\equiv \overline{B}_b(x) - \sum_{t=b}^{\ell} \binom{t}{b} \overline{Q}_t(x) \hat{S}^{(t-b,s-b)}(x) x^{u_t} && \mod x^{m_b} \end{aligned}$$

We should now choose $v_{t,b}$ such that the right-hand side still has degree less than ∇_b^B:

$$\begin{aligned} v_{t,b} &\leq \operatorname{pos}((\nabla_b^B - 1) - (\nabla_t^Q - 1) - u_t + 1) \\ &= \operatorname{pos}(\nabla_\ell^Q + (\ell-s)n - (\ell-b) \ - \ \nabla_t^Q \ - \ (\nabla_\ell^Q - \nabla_t^Q + (\ell-t)(n-1)) + 1) \\ &= \operatorname{pos}((\ell-s)n - (\ell-b) - (\ell-t)(n-1)) = \operatorname{pos}((t-s)n - (t-b-1)) \end{aligned}$$

For ease of analysis, we choose the (slightly suboptimal) $v_{t,b} = \mathrm{pos}((t-s)n-(t-1))$, and since this turns out not to depend on b, let $v_t = v_{t,b}$. We remind that for $t \le s$ then $v_t = 0$ and so $\hat{S}^{(t-b,s-b)}(x) = 0$. Remark that in spite of the Q's appearing on the right-hand side, the above equations still constitute a 2D key equation with $Q_0, \ldots, Q_\ell$ as a solution, since for a given Q, the right-hand sides simply *are* some polynomials of low degree.

Note now that in each congruence, then x^μ with $\mu = \min_t\{u_t + v_t\}$ divides the left-hand side and therefore also the right-hand side. That means we can divide the entire congruence, including modulus, with x^μ. Except when $s \ge k-1$ the following holds for μ:

$$\begin{aligned}\mu &= \min_t\{u_t + v_t\} \\ &= \min_t\{(\ell-t)(n-k) + \mathrm{pos}((t-s)n - t)\} \\ &= (\ell-s)(n-k) = u_s\end{aligned}$$

The last claim of the proposition is then that the degrees of the moduli, $m_b - \mu$, are in $O(sn)$, but since for real-valued t, the expression of μ has minimum at $t = s\frac{n}{n-1}$ we have

$$\begin{aligned}m_b - \mu &\ge \nabla^Q_\ell + (\ell-b)(n-1) - (\ell - s\tfrac{n}{n-1})(n-k) \\ &= s(n-\tau) + s\tfrac{n}{n-1}(n-k) - b(n-1)\end{aligned}$$ □

Example 3.39. *Consider the* $[250, 70, 181]$ *code from Example 3.5 on page 50 decoded up to* $\tau = 105$ *choosing* $s, \ell = (2, 4)$*. The many values become:*

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$
∇^Q_t	290	221	152	83	14
u_t	720	540	360	180	0
v_t	0	0	0	248	497

	$b=0$	$b=1$
∇^B_b	510	511
m_b	1010	761
$m_b - u_s$	650	401

Note in particular $\mu = u_s = 360$*, i.e. the number of terms that each congruence equation was reduced with by the reduction done in the proof of Proposition 3.38.* ♠

Theorem 3.40. *Assume* $s < k-1$*. Let* $Q = \sum_{t=0}^{\ell} Q_t(x)y^t$ *be a minimal* $(1, k-1)$*-weighted degree element in* $\mathcal{M}_{s,\ell}$*. Then* $(Q_0, \ldots, Q_\ell)$ *is a minimal solution to the 2D key equation of Type 1 with parameters*[1]

- $\rho = \ell + 1$ *and* $\sigma = s$*.*
- $G^{\mathrm{KE}}_j = x^{m_{j-1}-u_s}$ *for* $j = 1, \ldots, s$*.*
- $S^{\mathrm{KE}}_{i,j} = \binom{i-1}{j-1} S^{(i-j,s-j+1)}(x) x^{u_{i-1}+v_{i-1}-u_s}$ *if* $i \ge j$*, otherwise* $S^{\mathrm{KE}}_{i,j} = 0$*.*
- $\nu = 1$
- $N = s(n-\tau)$
- $\eta_i = (i-1)(k-1)$ *for* $i = 1, \ldots, \ell+1$

[1] To distinguish names occurring both here and in Section 2.5, we add $^{\mathrm{KE}}$ to the names of the latter.

- $w_j = sk - (j-1)$ *for* $j = 1, \dots, s$

where m_b, u_t, v_t *and* $S^{(t-b,s-b)}$ *are as in Proposition 3.38.*

Proof. We seek a 2D key equation such that $(Q_0, \dots, Q_\ell, \tilde{B}_0, \dots, \tilde{B}_{s-1})$ is a solution, so we map the definition of 2D key equation in Problem 2.31 on page 26, to Proposition 3.38. Only N and the weights are not obvious; however, we need only require N as an upper bound on the degrees of all Q_i and $\tilde{B}_j$, and then $\eta_i = N - \nabla^Q_{i-1}$ and $w_j = N - \nabla^{\tilde{B}}_{j-1}$ where $\nabla^{\tilde{B}}_{j-1} = \nabla^B_{j-1} - u_s$. Since

$$\begin{aligned}\nabla^B_b - u_s &= s(n-\tau) - \ell(k-1) + (\ell - s)n - (\ell - b) - (\ell - s)(n-k) \\ &= s(n-k-\tau) + b\end{aligned}$$

then N should exactly be $\nabla^Q_0 = s(n-\tau)$. The η_i and w_j then follow.

With this mapping, we see that $(Q_0, \dots, Q_\ell, \tilde{B}_0, \dots, \tilde{B}_{s-1})$ is a solution to the 2D key equation; since this is a minimal solution exactly when

$$\max_i \{\deg Q_{i-1} + \eta_i\} = \max_i \{\deg Q_{i-1} + i(k-1)\} = \deg_{1,k-1} Q$$

is minimal, the minimality concepts coincide. □

Remark. Remember that the approach of Theorem 3.40 still works perfectly well when $s \geq k-1$; in that case, one should replace u_s by μ in several of the expressions, and the analysis becomes somewhat more messy. ♦

Example 3.41. *Continuing Example 3.39, some of the values of Theorem 3.40 become ("x-pow" refers to the power of x in the expression of $S^{\mathrm{KE}}_{i,j}$):*

$N = 290$
$w_i = [140, 139]$

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
η_i	0	69	138	207	276
x-pow	360	180	0	68	137

To solve the equation, we will find a basis in $\Phi_{\nu,\bar{\boldsymbol{w}}}$-weighted weak Popov form of:

$$M = \left(\begin{array}{ccccc|cc} 1 & 0 & 0 & 0 & 0 & S^{(0,2)}(x)x^{360} & 0 \\ 0 & 1 & 0 & 0 & 0 & S^{(1,2)}(x)x^{180} & S^{(0,1)}(x)x^{180} \\ 0 & 0 & 1 & 0 & 0 & S^{(2,2)}(x) & 2S^{(1,1)}(x) \\ 0 & 0 & 0 & 1 & 0 & S^{(3,2)}(x)x^{68} & 3S^{(3,1)}(x)x^{67} \\ 0 & 0 & 0 & 0 & 1 & S^{(4,2)}(x)x^{137} & 4S^{(4,1)}(x)x^{137} \\ \hline 0 & 0 & 0 & 0 & 0 & x^{650} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x^{401} \end{array}\right)$$

$$\begin{aligned}\operatorname{maxdeg}(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) &= 790 \\ \Delta(\Phi_{\nu,\bar{\boldsymbol{w}}}(M)) &= 3216\end{aligned}$$

Compare the key properties of M with $A_{2,4}$ from Example 3.28 on page 62. ♠

Worst-case complexity of solving the Q-finding 2D key equation of Theorem 3.40

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^3 s^2 n^2$	$\ell^3 s^2 n^2$
Alekhnovich	$\mathrm{M}(\ell)\mathrm{P}(\ell sn)\log(\ell sn) + \ell^2\mathrm{P}(sn)$	$\ell^4 sn \log(n)^{2+o(1)}$
GJV	$\mathrm{M}(\ell)\mathrm{P}(sn)\log(\ell sn)^{O(1)}$	$\ell^3 sn \log(n)^{O(1)}$
D–D	$\ell^3 s^2 n^2$	$\ell^3 s^2 n^2$

Table 3.3: For relaxation, we have used the same rules as in Table 3.1 on page 63.

Proposition 3.42. *The worst-case complexity of finding a satisfactory interpolation polynomial Q by solving the 2D key equation of Theorem 3.40 is as in Table 3.3, for various choices of module minimisation algorithm.*

Proof. Follows from Table 2.4 on page 42 after estimating the measures of the 2D key equation. Observe that $\deg G_j^{\mathrm{KE}} \in O(sn)$ since $m_{j-1} - u_s \in O(sn)$ by Proposition 3.38. Also clearly $N, \eta_i, w_J \in O(sn)$ for all i, j. Now we have

$$\begin{aligned} \gamma = \operatorname{maxdeg}(\Phi_{1,\bar{\boldsymbol{w}}}(M)) &= \max\{\eta_1, \dots, \eta_\ell, \deg G_1^{\mathrm{KE}} + w_1, \dots, \deg G_\rho^{\mathrm{KE}} + w_\rho\} \\ &\in O(sn) \\ \delta = \Delta(\Phi_{1,\bar{\boldsymbol{w}}}(M)) < \rho\gamma &\in O(\ell sn) \end{aligned}$$

□

Remark. Note that the relaxed complexities of Table 3.3 are all the same as in Table 3.1 on page 63, meaning that in a strict asymptotic sense, the method described here is as fast as that of Section 3.2. Looking at the expressions before the relaxation, we might be tempted to conclude that the latter is actually fastest, but a more precise analysis would be necessary. Remember also the warnings of Section 1.2. ◆

Remark. There is no need to reverse the order of the polynomials as we did in Proposition 3.38 in order to arrive at a 2D key equation; however, without the flipping, it does not seem possible to perform a reduction such as the one we did by x^{μ_b}. As previously mentioned, this would cause the complexity estimates to go up, since nothing better than $\gamma \in O(\ell n)$ could be stated in general.

The original derivation in [ZGA11] has some issues; referring to their labelling, it is from equation (33) to (35) and (36), and I have discussed this with one of the authors, Alexander Zeh. In particular, a reduction such as the one by x^{μ_b} is not explained, but seems to have been applied in (36), and the extracted part of $S_\diamond^{(a,b)}(x)$ are indexed wrongly. Alexander Zeh's PhD thesis [Zeh13] contains a new, more verbose, description of the derivation, which seems to follow the same line of arguments as here. ◆

3.4.1 The Roth–Ruckenstein reduction in the Sudan case

For $s = 1$, i.e. the Sudan case, then $\sigma = 1$ and there is only one equation of the form

$$\sum_{t=0}^{\ell} \overline{Q}_t(x) S^{(t,1)}(x) x^{u_t+v_t-\mu} \equiv \tilde{B}_0(x) \mod x^{m_0-\mu} \tag{3.10}$$

If we lower the degree of the modulus by $m_0-(u_0+v_0) = n-\tau$, then we completely eliminate Q_0 from the equation (and since $(m_0-\mu)-(m_0-u_0) > \nabla_0^{\tilde{B}} = \nabla_0^B - \mu$, the right-hand side still has degree less than the resulting modulus). This lowering will in general reduce the information in the key equation: regarding it as a linear equation in the coefficients of the Q_t (as in the proof of Proposition 2.39 on page 33), we are losing $n-\tau$ equations. Coincidentally, $\nabla_0^Q = n-\tau$ which means that we are also getting rid of $n-\tau$ variables! Therefore, the reduced linear system must have exactly the same space of solutions. We can therefore solve the reduced key equation for $Q_1, \dots, Q_\ell$ and then afterwards find Q_0, e.g. by Lagrangian interpolation: since $Q(\alpha_i, r_i') = 0$ for $i = 1, \dots, n$ then $Q_0(\alpha_i)$ must be something known once $Q_1, \dots, Q_\ell$ have been established. Expanding and simplifying the values of the parameters in the instance $s = 1$, we end up with the reduced key equation for $Q_1, \dots, Q_\ell$:

$$\sum_{t=1}^{\ell} \overline{Q}_t(x) S^{(t,1)}(x) x^{(t-1)(k-1)} \equiv \tilde{B}_0(x) \mod x^{n-k} \tag{3.11}$$

with $\deg \tilde{B}_0 < n-k-\tau$. The module to minimise in order to solve this 2D key equation is then a weighted variant of

$$M_{\mathrm{RR}} = \left(\begin{array}{ccc|c} 1 & & & S^{(1,1)} \\ & 1 & & S^{(2,1)} x^{k-1} \\ & & \ddots & \vdots \\ & & 1 & S^{(\ell,1)} x^{(\ell-1)(k-1)} \\ \hline \multicolumn{3}{c|}{0} & x^{n-k} \end{array}\right) \tag{3.12}$$

This reduction was used by Roth and Ruckenstein [RR00] in their paper giving the key equation approach for the Sudan case. Unfortunately, it does not seem to work for $s > 1$: $\nabla_0^Q = s(n-\tau)$ but since $m_0 - \mu - \nabla_0^{\tilde{B}} < s(n-\tau)$ reducing the degree of the modulus by $s(n-\tau)$ does not remove that many equations. For $t > 1$ each Q_t appears in more than one of the key equations, and so also seem to be too entangled with each other within the linear system to be removable.

Example 3.43. *Consider again the [250, 40, 211] from Example 3.20 on page 59 and choosing $\ell = 3$ to decode 129 errors. The parameters become*

$$\nabla_t^Q = [122, 83, 44, 5] \qquad (t-1)(k-1) = [0, 39, 78] \qquad n-k-\tau = 82$$
$$\eta_i = [39, 78, 117] \qquad w_1 = 40$$
$$\operatorname{maxdeg}(\Phi_{\nu,\vec{w}}(M_{\mathrm{RR}})) = 250 \qquad \Delta(\Phi_{\nu,\vec{w}}(M_{\mathrm{RR}})) = 669$$

Compared with the 2D key equation, η_i runs for $i = 2, 3, 4$ since the $\overline{Q}_0$ equation was eliminated. ♠

3.5 Guruswami–Sudan decoding Hermitian codes

The Guruswami–Sudan algorithm generalise to a large class of AG codes, and this was even done in the original publication [GS99] based on Shokrollahi and Wasserman's generalisation of the original Sudan algorithm [SW99]. In this section we will show how this works on one-point Hermitian codes; they are simple, have excellent parameters, and are probably first in line as an AG alternative to GRS codes. Still, the results here should generalise rather easily to larger classes of codes, see also Section 3.6.

The main result of the section is a description of how to find a satisfactory Q-polynomial in a fast way, analogous to what we did in Section 3.2. As before, this is strongly inspired by existing work: both Lee and O'Sullivan as well as Beelen and Brander have extended their algorithms for GRS codes to Hermitian or more general AG codes. Recall that these two approaches were basically that of Section 3.2 using the Mulders–Storjohann, respectively the Alekhnovich algorithm for module minimisation. Recall also that asymptotically, using the GJV instead beats these two approaches; exactly the same will be the case in this section. However, to be able to achieve this better performance, we will propose a slightly different way to embed the problem of the function field into one over $\mathbb{F}[x]$ than what was done previously. The result is an algorithm with the best known complexity for list decoding Hermitian codes.

We will here only focus on finding Q. In particular, we will not discuss parameter choices, and we will not discuss root-finding. For the former, one could perform a similar type of analysis as what we did in Section 3.1.2, but just as for the case of GRS codes, I am (surprisingly) not aware of anyone having done this. For the latter, Wu and Siegel [WS01] describe how to generalise the Roth–Ruckenstein root-finding algorithm to polynomials over algebraic function fields. I am not aware of a generalisation of Alekhnovich's D&C speedup for the GRS case [Ale05], but one could suspect that it is possible.

The exposition will be to the point, and so requires a certain familiarity with algebraic function fields; however, the necessary concepts are all from introductory parts of the theory. For instance, Chapters 1 and 2 of Stichtenoth's book [Sti09] should cover it.

3.5.1 Preliminary concepts and the codes

Let q be some prime power, and consider the curve $\mathcal{H}$ over $\mathbb{F}_{q^2}$ defined by the following polynomial in X, Y:

$$\mathcal{H}(X,Y) = Y^q + Y - X^{q+1}$$

$\mathcal{H}$ is the Hermitian curve, and it is absolutely irreducible. Let $F = \mathbb{F}_{q^2}(x,y)$ be the algebraic function field achieved by extending $\mathbb{F}_{q^2}(x)$ with a variable y satisfying the relation $\mathcal{H}(x,y) = 0$.

There are certain basic facts about F which we will need:

Proposition 3.44. *The genus of $\mathcal{H}$ is $g = \frac{1}{2}q(q-1)$. F has q^3+1 rational places, which we will denote $\mathcal{P} = \{P_1, \ldots, P_{q^3}, P_\infty\}$. The place P_∞ is "the place at infinity" and the only rational place which has a pole at either x or y (it has both). It has ramification index q over $\mathbb{F}_{q^2}(x)$. Furthermore define*

$$\mathfrak{R} = \bigcup_{i=0}^{\infty} \mathcal{L}(iP_\infty)$$

Then $\mathfrak{R} = \mathbb{F}_{q^2}[x,y]$.

Let $\mathcal{P}^\star = \mathcal{P} \setminus \{P_\infty\}$. We can decompose $\mathcal{P}^\star$ into q^2 disjoint subsets each of size q, indexed by the $\alpha \in \mathbb{F}_{q^2}$, and such that $P_{\alpha,1}, \ldots, P_{\alpha,q}$ all lie above the zero of $x - \alpha \in \mathbb{F}_{q^2}(x)$. In particular, this means that $(x-\alpha) = \sum_{j=1}^{q} P_{\alpha,j} - qP_\infty$.

Proof. We will not give the proof here but refer to the literature. Genus and rational points can be found in any text dealing with the Hermitian curve. The fact on $\mathfrak{R}$ is well-known for function fields over "nice curves"; that as well as the facts on ramification of the affine places are in [Bra10, Section 2.3]. □

That $\mathfrak{R} = \mathbb{F}_{q^2}[x,y]$ is extremely helpful since all these functions can then be described by polynomials. For brevity, we will write ∞P_∞ in Riemann–Roch-space definitions to mean the union of all the Riemann–Roch spaces where ∞P_∞ is replaced by iP_∞ for $i = 0, 1 \ldots$. For instance, $\mathfrak{R} = \mathcal{L}(\infty P_\infty)$.

We should furthermore keep in mind the relation $\mathcal{H}(x,y) = 0$. When we need to consider an element $f \in \mathfrak{R}$ concretely as a polynomial in x and y, we will always consider it in a form where $\mathcal{H}(x,y) = 0$ has been used to maximally reduce the y-degree of f to less than q; we will call this "reducing by $\mathcal{H}$". We will measure elements of $\mathfrak{R}$ by their pole order in P_∞; when elements in $\mathfrak{R}$ have been reduced by $\mathcal{H}$, this takes on a particularly simple form:

Definition 3.45. Let $\deg_{\mathcal{H}} : \mathfrak{R} \mapsto \mathbb{N}_0 \cup \{-\infty\}$ be given as $\deg_{\mathcal{H}}(p) = -v_{P_\infty}(p)$ for $p \neq 0$ and $\deg_{\mathcal{H}}(0) = -\infty$, where $v_P(\cdot)$ is the valuation of a function at the place P. For non-zero input, this is also equal to the $\mathbb{F}_{q^2}$-linear closure of

$$\deg_{\mathcal{H}}(x^i y^j) = \deg_{q,q+1}(x^i y^j) = qi + (q+1)j$$

when $j < q$.

Note that all monomials $x^i y^j$ with $j < q$ have different $\deg_{\mathcal{H}}$. Therefore, $\deg_{\mathcal{H}}$ induces a term ordering $\leq_{\mathcal{H}}$ on $\mathbb{F}_{q^2}[x, y]$ such that $x^{i_1} y^{j_1} \leq_{\mathcal{H}} x^{i_2} y^{i_2}$ if and only if $\deg_{\mathcal{H}}(x^{i_1} y^{j_1}) \leq \deg_{\mathcal{H}}(x^{i_2} y^{j_2})$ or $i_1 = i_2 \wedge j_1 = j_2$. This means that we can speak of leading monomial, $\mathrm{LM}_{\mathcal{H}}(\cdot)$, and leading coefficient, $\mathrm{LC}_{\mathcal{H}}(\cdot)$, for elements of $\mathcal{R}$.

We will also need two easy technical lemmas:

Lemma 3.46 ([Bra10, Proposition 2.2])**.** *For any non-zero $h \in F$ then*

$$\mathcal{L}(-(h) + \infty P_\infty) = h\mathcal{R} \tag{3.13}$$

Proof. If $f \in h\mathcal{R}$ then $(f) \geq (h) - tP_\infty$ for some t, but then $f \in \mathcal{L}(-(h) + tP_\infty)$ which is clearly in the left-hand side of (3.13).

On the other hand, if $g \in \mathcal{L}(-(h) + \infty P_\infty)$ then for some t we have $(g) \geq (h) - tP_\infty$. Therefore $(g/h) \geq -tP_\infty$ so $g/h \in \mathcal{R}$. Then clearly $g \in h\mathcal{R}$. □

Lemma 3.47. *For any $m \in \mathbb{Z}_+$, there are at least $m - g$ distinct monomials of the form $x^i y^j$, $j < q$ such that $\deg_{\mathcal{H}}(x^i y^j) < m$.*

Proof. The statement translates simply to $\dim \mathcal{L}((m-1)P_\infty) \geq m - g$, which is exactly Riemann's Theorem, see e.g. [Sti09, Theorem 1.4.17]. □

Let us now formally introduce the class of codes we will be able to decode. Often in the literature, only Hermitian codes where *all* q^3 affine places $\mathcal{P}^\star$ are used are considered, but we will support a slightly larger family.

Definition 3.48. Choose some $n \leq q^3$ with $n \equiv 0 \mod q$, and choose n/q distinct elements of $\mathbb{F}_{q^2}$, $\alpha_1, \ldots, \alpha_{n/q}$. Let $D = \sum_{i=1}^{n/q} \sum_{j=1}^{q} P_{\alpha_i, j}$, and for brevity label $\mathrm{supp} D$ as $P_1, \ldots, P_n$ in some arbitrary manner. Let m be an integer satisfying $2g - 2 < m < n$. Then the corresponding Hermitian code over $\mathbb{F}_{q^2}$ is defined as

$$\mathcal{C} = \left\{ \big(f(P_1), \ldots, f(P_n)\big) \mid f \in \mathcal{L}(mP_\infty) \right\}$$

Note that $\mathcal{L}(mP_\infty) \subset \mathcal{R}$, so all the f we need to evalute to obtain $\mathcal{C}$ are polynomials in x and y satisfying $\deg_{\mathcal{H}} f \leq m$. The basic parameters of the codes are almost completely known:

Proposition 3.49. *In the context of Definition 3.48, $\mathcal{C}$ is an $[n,\ k,\ \geq d]$ code where*

$$k = m - g + 1 \qquad\qquad d = n - m$$

Proof. The proposition is standard for any AG code, see e.g. [Sti09, Theorem 2.2.2] □

Remark. When $n = q^3$, i.e. we are evaluating at all affine points, then the *exact* minimum distance is known: Stichtenoth showed that it is exactly d as above whenever $m \leq n - q^2$ [Sti88], while the remaining cases were determined by Yang and Kumar and shown to be slightly better for some of the m [YK92].

Note also that the construction of the codes work perfectly well with $m \leq 2g-2$, but that for these codes, the dimension and minimum distance are slightly less straightforward. To describe clearly for which parameters our decoder has the claimed performance, we have opted to leave the restriction $m > 2g-2$ in the definition. ♦

In the following section, we will consider dealing with a particular choice of such an Hermitian code, and use all these introduced variables $n, k, P_i, \mathcal{C}$, etc.

As a last tool before we begin, we will also need Lagrangian interpolation over the evaluation points of a considered Hermitian code, i.e. given $\gamma_i \in \mathbb{F}_{q^2}$ for $i = 1, \ldots, n$ then find some $p \in \mathcal{R}$ such that $p(P_i) = \gamma_i$ for all i. It is easy to see such a function must exist: for each i, the requirement specifies a linear equation in the coefficients over p seen as an element of $\mathbb{F}_{q^2}[x, y]$, so by Lemma 3.47 on page 80 there must exist one with $\deg_{\mathcal{H}}$ less than $n + g + 1$. However, for correctness, it will turn out that any interpolation function will suffice for us. Since it is slow to solve a linear system of equations, it is beneficial to have a closed formula though this might yield a function of slightly suboptimal order. The following lemma is inspired by a similar result from [LO09]:

Lemma 3.50. *Let (α_i, β_i) be the coordinates corresponding to P_i for $i = 1, \ldots, n$. Let $A = \{\alpha_i\}_i$ and $B_i = \{\beta_j \mid \alpha_j = \alpha_i\}_j$ for $i = 1, \ldots, n$. For $\gamma_i \in \mathbb{F}_{q^2}$ for $i = 1, \ldots, n$ then $p \in \mathcal{R}$ given by*

$$p(x, y) = \sum_{i=1}^{n} \gamma_i \prod_{\alpha \in A \setminus \{\alpha_i\}} \frac{x - \alpha}{\alpha_i - \alpha} \prod_{\beta \in B_i \setminus \{\beta_i\}} \frac{y - \beta}{\beta_i - \beta}$$

satisfies $p(P_i) = \gamma_i$ for $i = 1, \ldots, n$ and $\deg_{\mathcal{H}} p < n + 2g$

Proof. Clearly, the given $p \in \mathcal{R}$. For the ith term, the first product ensures evaluation is 0 at any $P_j, j \neq i$ for which $\alpha_i \neq \alpha_j$, while the second product ensures the same for j where $\alpha_i = \alpha_j$ but $\beta_i \neq \beta_j$. We also see that at P_i, the evaluation must be γ_i, proving the first claim.

For the order, recall from the restriction on the choice of P_i in Definition 3.48 on page 80, that we must have $|A| = n/q$ and $|B_i| = q$ for each i. Therefore $\deg_x(p) \leq n/q - 1$ and $\deg_y(p) \leq q - 1$, so $\deg_{\mathcal{H}}(p) \leq q(n/q-1) + (q+1)(q-1)$. □

3.5.2 Decoding

We will be working with elements of $\mathcal{R}[z]$, i.e. the univariate polynomial ring over $\mathcal{R}$; this is going to take the place of $\mathbb{F}[x][y]$ of the GRS case. We naturally extend our existing zoo of degree functions with $\deg_{\mathcal{H},w}$ for any $w \in \mathbb{R}$, so that some $Q = \sum_{t=0}^{\deg_z Q} Q_t(x, y) z^t \in \mathcal{R}[z]$ has $\deg_{\mathcal{H},w} Q = \max_t \{\deg_{\mathcal{H}} Q_t + tw\}$. Extend also for such a Q the coefficient-selecting notation $Q_{[t]}$ to mean $Q_{[t]} = Q_t(x, y) \in \mathcal{R}$.

Definition 3.51. A polynomial $Q \in \mathit{Я}[z]$ has a zero $(P, z_0) \in \mathcal{P}^\star \times \mathbb{F}_{q^2}$ with multiplicity s if Q can be written as $\sum_{j+h\geq s} \gamma_{j,h}\phi^j(z-z_0)^h$ for some $\gamma_{j,h} \in \mathbb{F}_{q^2}$, where ϕ is a local parameter for P.

Theorem 3.52 (Guruswami–Sudan for Hermitian codes)**.** *Let $s, \ell, \tau \in \mathbb{Z}_+$ be given. If $Q \in \mathit{Я}[z]$ with $Q = \sum_{t=0}^{\ell} Q_i(x,y)z^t \neq 0$ satisfying*

1. *Q has a zero at (P_i, r_i) with multiplicity s for $i = 1, \dots, n$.*
2. *$\deg_{\mathcal{H}} Q_i < s(n-\tau) - im$.*

and if $|\mathcal{E}| \leq \tau$ then $Q(f) = 0$.

Proof. Consider $\hat{Q} = Q(f) \in \mathit{Я}$. Clearly $\deg_{\mathcal{H}} \hat{Q} < s(n-\tau)$ by Point 2. By Point 1 then for each i, Q can be written as $\sum_{j+h\geq s} \gamma_{j,h}\phi_i^j(z-r_i)^h$ for some $\gamma_{j,h} \in \mathbb{F}_{q^2}$, where ϕ_i is a local parameter for P_i. That means $\hat{Q} = \sum_{j+h\geq s} \gamma_{j,h}\phi_i^j(f-r_i)^h$. If $i \notin \mathcal{E}$ then both $(f-r_i)$ and ϕ_i have a zero at P_i, so the above implies that $\hat{Q} \in \mathcal{L}(-sP_i + \infty P)$. Connecting this with $\deg_{\mathcal{H}} \hat{Q}$ found above, we thus get

$$\hat{Q} \in \bigcap_{i\notin\mathcal{E}} \mathcal{L}\big(-sP_i + (s(n-\tau)-1)P_\infty\big) = \mathcal{L}\Big(-s\sum_{i\notin\mathcal{E}} P_i + (s(n-\tau)-1)P_\infty\Big)$$

But $\deg\big(-s\sum_{i\notin\mathcal{E}} P_i + (s(n-\tau)-1)P_\infty\big) < 0$ if $|\mathcal{E}| \leq \tau$ so the Riemann–Roch space contains only 0, whence $\hat{Q} = 0$. □

Remark. An analogous statement holds for much more general AG codes, though $\mathit{Я}$ of course needs to be defined properly; the proof also carries over more or less unchanged. See e.g. [GS99] or the clear description of [BH08a]. ◆

One can find a satisfactory Q by solving a linear system of equations in the $\mathbb{F}_{q^2}$-coefficients for its $x^iy^jz^h$-monomials, and one can ensure that this system will have a non-zero solution by satisfying a certain equation in the parameters, completely analogous to Proposition 3.4 on page 49. The resulting equation can be analysed for determining the maximal τ and corresponding choices of s and ℓ. We are not going to perform that analysis; see e.g. [GS99] for a basic, asymptotically satisfactory analysis. The summary of the analysis is that choices of s and ℓ exists ensuring the existence of a satisfactory Q as long as $\tau < n - \sqrt{n(n-d)}$, i.e. the asymptotic Johnson bound again!

Remark. An interesting oddity when decoding AG codes is that in the case $\ell = s = 1$, one does not achieve half the minimum distance! With a rudimentary analysis, one is only guaranteed of success when $\tau \leq \frac{d-1}{2} - g$. By careful inspection of the linear system of equations, this turns out to be improvable to $\frac{d-1}{2} - \frac{g}{2}$ [Bee13]; but still not half the minimum distance. We will meet a similar deficiency when power decoding Hermitian codes in Section 4.4. ◆

3.5.3 Finding Q in an explicit module

We will only concern ourselves with the problem of finding Q; our approach is completely analogous to that of Section 3.2. We will again assume $s \leq \ell$, and with the proper analysis of choices of s and ℓ, one can realise that this is no real restriction.

Definition 3.53. Let $\mathcal{M}^{\mathcal{H}}_{s,\ell} \subset Я[z]$ denote the set of all $Q \in Я[z]$ such that Q has a zero of multiplicity s at (P_i, r_i) for $i = 1, \ldots, n$, and $\deg_z Q \leq \ell$.

Finding a $Q \in Я[z]$ for satisfying the requirements of Theorem 3.52 is then the same as finding an element in $\mathcal{M}^{\mathcal{H}}_{s,\ell}$ with low enough $\deg_{\mathcal{H},m}$.

To continue, we first need functions $R, G \in Я$, analogous to those we had for GRS codes in Definition 3.22 on page 60.

Definition 3.54.

$$G = \prod_{i=1}^{n/q} (x - \alpha_i) \qquad\qquad R \neq 0 : \; R(P_i) = r_i \quad \forall i = 1, \ldots, n$$

Note that by Proposition 3.44 on page 79, we must have $(G) = \sum_{i=1}^{n} P_i - nP_\infty$, i.e. analogous to the case for GRS codes we have $G(P_i) = 0$ for $i = 1, \ldots, n$. For R, any non-zero function in $Я$ satisfying the interpolation constraints will do; we can either solve the linear system of equations in its coefficients, or we can use the explicit formula of Lemma 3.50 on page 81.

The following two results perfectly mimic Lemma 3.23 and Theorem 3.24 on page 60, and the proofs are almost trivial rewrites of those results'.

Lemma 3.55. *Let $Q \in \mathcal{M}^{\mathcal{H}}_{s,\ell}$ with $\deg_z Q = t < s$. Then $G^{s-t} \mid Q_{[t]}$ as a division over $Я$.*

Proof. For each i, Q has a zero (P_i, r_i) with multiplicity s, which means that $Q = \sum_{j+h \geq s} \gamma_{j,h} \phi_i^j (z - r_i)^h$ for some $\gamma_{j,h} \in \mathbb{F}_{q^2}$, where ϕ_i is any local parameter for P_i. Since $\deg_z Q = t < s$ then $h \leq t$, which means $j \geq s - t$.

Therefore, $Q = \phi_i^{s-t} \sum_{j+h \geq s} \gamma_{j,h} \phi_i^{j-(s-t)} (z - r_i)^h = \phi_i^{s-t} \sum_{h=0}^{t} q_h(x, y) z^h$ for some $q_h \in F$. The q_h must be power series in ϕ_i, so they each have poles only at P_∞, i.e. $q_h \in Я$; therefore the entire sum must be in $Я[z]$, i.e. ϕ_i^{s-t} divides Q as a division over $Я$. Now, it is easy to show that $\phi_i = x - \alpha_i$ is a valid choice as local parameter for P_i; collecting for all different i, we therefore have $G^{s-t} \mid Q$, immediately implying the sought. □

Theorem 3.56. *The module $\mathcal{M}^{\mathcal{H}}_{s,\ell}$ is generated as an $Я$-module by the $\ell + 1$ polynomials $H^{(i)} \in Я[z]$ given by*

$$H^{(t)}(z) = G^{s-t}(z - R)^t, \qquad \text{for } 0 \leq t \leq s,$$
$$H^{(t)}(z) = z^{t-s}(z - R)^s, \qquad \text{for } s < t \leq \ell.$$

Proof. It is easy to see that each $H^{(t)} \in \mathcal{M}^{\mathcal{H}}_{s,\ell}$ since both G and $z-R$ have a zero of multiplicity one at (P_i, r_i) for $i = 1, \ldots, n$, and that G and $z - R$ divide $H^{(t)}$ with total power s for each t.

To see that any element of $\mathcal{M}^{\mathcal{H}}_{s,\ell}$ can be written as an $\mathcal{R}$-combination of the $H^{(t)}$, let $Q(z)$ be some element of $\mathcal{M}^{\mathcal{H}}_{s,\ell}$. Then the polynomial $Q^{(\ell-1)}(z) = Q - Q_{[\ell]}(x,y)H^{(\ell)}$ has z-degree at most $\ell - 1$. Since both Q and $H^{(\ell)}$ are in $\mathcal{M}^{\mathcal{H}}_{s,\ell}$, so must $Q^{(\ell-1)}$ be in $\mathcal{M}^{\mathcal{H}}_{s,\ell}$. Since $H^{(t)}$ has z-degree t and $H^{(t)}_{[t]}(x,y) = 1$ for $t = \ell, \ell-1, \ldots, s$, we can continue reducing this way until we reach a $Q^{(s-1)}(z) \in \mathcal{M}^{\mathcal{H}}_{s,\ell}$ with z-degree at most $s-1$. From then on, we have $H^{(t)}_{[t]}(x,y) = G^{s-t}(x)$, but by Lemma 3.55, we must also have $G(x) \mid Q^{(s-1)}_{[s-1]}(x,y)$, so we can also reduce by $H^{(s-1)}$. This can be continued with the remaining $H^{(t)}$, eventually reducing the remainder to 0. □

The above theorem gives us a basis in $\mathcal{R}[z]$ of the $\mathcal{R}$-module $\mathcal{M}^{\mathcal{H}}_{s,\ell}$, and we are seeking the minimal $\deg_{\mathcal{H},m}$-weighted element in this module. We need to project this module, its basis and the weighted degree into $\mathbb{F}_{q^2}[x]$ in some sensible manner to be able to use the tools of Chapter 2 to solve this problem. This turns out to conceptually be rather straightforward: we expand elements of $\mathcal{R} = \mathbb{F}_{q^2}[x,y]$ into vectors over $\mathbb{F}_{q^2}[x]$ by their $\mathbb{F}_{q^2}[x]$-coefficients when regarding them as polynomials in y, using the representation which has been reduced by $\mathcal{H}$, and with a few further tricks we obtain a module over $\mathbb{F}_{q^2}[x]$. To write this down formally requires introducing a bit of notation, though.

Firstly, introduce $\Upsilon : \mathcal{R} \mapsto \mathbb{F}_{q^2}[x]^q$ by for any element of $\mathcal{R}$, say $g = \sum_{i=0}^{q-1} y^i g_i(x)$, then $\Upsilon(g) = (g_0, \ldots, g_{q-1})$. As we have previously noted, any element of $\mathcal{R}$ *can* be written such that the y-degree is at most q and in only one way.

Of course, we can also consider vectors over $\mathbb{F}_{q^2}[x]$ of length more than q, and this can be useful as intermediate calculations. In particular, a multiplication $g \cdot h$ for $g, h \in \mathcal{R}$ where $g = \sum_{i=0}^{q-1} g_i(x)y^i$ and $h = \sum_{i=0}^{q-1} h_i(x)y^i$ can be represented as

$$\begin{pmatrix} g_0 & g_1 & \ldots & g_{q-1} \end{pmatrix} \begin{pmatrix} h_0 & h_1 & \ldots & h_{q-1} & & & 0 \\ & h_0 & h_1 & \ldots & h_{q-1} & & \\ & & \ddots & & & \ddots & \\ 0 & & & h_0 & h_1 & \ldots & h_{q-1} \end{pmatrix} \tag{3.14}$$

The result will be a vector $(p_0, \ldots, p_{2q-2}) \in \mathbb{F}_{q^2}[x]^{2q-1}$ such that $g \cdot h = \sum_{i=0}^{2q-2} p_i(x)y^i$. Denote by $\Pi(h)$ the matrix of the above form, for any $h \in \mathcal{R}$. Reducing $\sum_{i=0}^{2q-2} q_i(x)y^i$ by $\mathcal{H}$ down to y-degree less than q becomes the result of the linear transformation

$$\begin{pmatrix} p_0 & p_1 & \ldots & p_{2q-2} \end{pmatrix} \left(\begin{array}{cccc} \multicolumn{4}{c}{I_{q\times q}} \\ \hline x^{q+1} & -1 & & 0 \\ & \ddots & \ddots & \\ 0 & & x^{q+1} & -1 \\ & & & x^{q+1} \end{array} \right) \tag{3.15}$$

Where $I_{q\times q}$ is the $q \times q$ identity matrix. Denote the matrix in the above product by Ξ. With this notation then we can write

$$\Upsilon(g \cdot h) = \Upsilon(g)\Pi(h)\Xi \tag{3.16}$$

Let $\mathcal{R}[z]_\ell$ be polynomials of z-degree at most ℓ; then we also introduce $\Upsilon_z \colon \mathcal{R}[z]_\ell \mapsto \mathbb{F}_{q^2}[x]^{(\ell+1)q}$, by for some $Q \in \mathcal{R}[z]_\ell$, then $\Upsilon_z(Q) = \big(\Upsilon(Q_{[0]}) \mid \ldots \mid \Upsilon(Q_{[\ell]})\big)$.

Proposition 3.57. *Let $A^{\mathcal{H}}_{s,\ell} \in \mathbb{F}_{q^2}[x]^{(q(\ell+1))\times(q(\ell+1))}$ be given by*

$$A^{\mathcal{H}}_{s,\ell} = \left[\begin{array}{c|c|c|c} \Pi(H^{(0)}_{[0]})\Xi & \mathbf{0} & \ldots & \mathbf{0} \\ \hline \Pi(H^{(1)}_{[0]})\Xi & \Pi(H^{(1)}_{[1]})\Xi & \ldots & \mathbf{0} \\ \hline \vdots & & \ddots & \vdots \\ \hline \Pi(H^{(\ell)}_{[0]})\Xi & \Pi(H^{(\ell)}_{[1]})\Xi & \ldots & \Pi(H^{(\ell)}_{[\ell]})\Xi \end{array}\right]$$

$\mathcal{M}^{\mathcal{H}}_{s,\ell}$ is in bijection with the $\mathbb{F}_{q^2}[x]$ row space of $A^{\mathcal{H}}_{s,\ell}$ through the map Υ_z. Furthermore, let $\Upsilon_z(Q)$ be the element in the $\mathbb{F}_{q^2}[x]$ row space of $A^{\mathcal{H}}_{s,\ell}$ with minimal $\Phi_{1,\boldsymbol{w}}$-weighted degree, where $\boldsymbol{w} = (w_{0,0}, \ldots, w_{0,q-1}, w_{1,0}, \ldots, w_{\ell,q-1})$ and

$$w_{t,j} = \lfloor q^{-1}(tm + (q+1)j) - \varepsilon + 1 \rfloor$$

where $\varepsilon = q^{-1}s(n-\tau) - \lfloor q^{-1}s(n-\tau) \rfloor$. Then $\deg_{\mathcal{H},m} Q < s(n-\tau)$.

Proof. Consider some $Q(z) \in \mathcal{M}^{\mathcal{H}}_{s,\ell}$; by Theorem 3.52 we can find $p_t \in \mathcal{R}$ such that $Q(z) = \sum_{t=0}^{\ell} p_t H^{(t)}(z)$. Clearly then $Q_{[h]} = \sum_{t=0}^{\ell} p_t H^{(t)}_{[h]}$. Now expand this equation using Υ:

$$\Upsilon(Q_{[h]}) = \sum_{t=0}^{\ell} \Upsilon(p_t H^{(t)}_{[h]}) = \sum_{t=0}^{\ell} \Upsilon(p_t)\Pi(H^{(t)}_{[h]})\Xi$$

Since $\Upsilon_z(Q) = \big(\Upsilon(Q_{[0]}) \mid \ldots \mid \Upsilon(Q_{[\ell]})\big)$, we get

$$\begin{aligned} \Upsilon_z(Q) &= \sum_{t=0}^{\ell} \Upsilon(p_t)\Big(\Pi(H^{(t)}_{[0]})\Xi \mid \ldots \mid \Pi(H^{(t)}_{[\ell]})\Xi\Big) \\ &= \sum_{t=0}^{\ell} \Upsilon(p_t)\Big(\Pi(H^{(t)}_{[0]})\Xi \mid \ldots \mid \Pi(H^{(t)}_{[t]})\Xi \mid \mathbf{0} \mid \ldots \mid \mathbf{0}\Big) \\ &= \big(\Upsilon(p_0) \mid \ldots \mid \Upsilon(p_\ell)\big)A^{\mathcal{H}}_{s,\ell} \end{aligned}$$

where the second step is due to $\deg_z H^{(t)} = t$. These steps are reversible, so also any element in the $\mathbb{F}_{q^2}[x]$ row space of $A^{\mathcal{H}}_{s,\ell}$ must be the $\Upsilon_z(\cdot)$-image of an element in $\mathcal{M}^{\mathcal{H}}_{s,\ell}$.

For the claim on the minimality, write $Q = \sum_{t=0}^{\ell}\sum_{h=0}^{q-1} Q_{t,h}(x)y^h z^t$, for some

$Q_{t,h} \in \mathbb{F}_{q^2}[x]$. We do some calculation

$$\begin{aligned}
&\deg(\Phi_{1,\boldsymbol{w}}(\Upsilon_z(Q))) = \max_{t,h}\{w_{t,h} + \deg Q_{t,h}\} \\
&= \max_{t,h}\left\{\lfloor q^{-1}((q+1)j + tm) - \varepsilon + 1\rfloor + \deg Q_{t,h}\right\} \\
&= \left\lfloor \max_{t,h}\left\{q^{-1}((q+1)j + tm) + \deg Q_{t,h}\right\} - \varepsilon + 1 \right\rfloor \\
&= \lfloor q^{-1}\deg_{\mathcal{H},m} Q - \varepsilon + 1\rfloor
\end{aligned}$$

Therefore, $\deg_{\mathcal{H},m} Q < s(n-\tau) \iff \deg(\Phi_{1,\boldsymbol{w}}(\Upsilon_z(Q))) < \lfloor q^{-1}s(n-\tau)+1\rfloor$, by the definition of ε.

Since we know there exists a $Q \in \mathcal{M}^{\mathcal{H}}_{s,\ell}$ with $\deg_{\mathcal{H},m} Q < s(n-\tau)$, then $\Upsilon_z(Q)$ must have $\Phi_{1,\boldsymbol{w}}$-weighted degree less than $\lfloor q^{-1}s(n-\tau)+1\rfloor$, and so surely the minimal weighted-degree in the row space of $A^{\mathcal{H}}_{s,\ell}$ is at most this. This minimal element's corresponding polynomial in $\mathcal{M}^{\mathcal{H}}_{s,\ell}$ must then have $\deg_{\mathcal{H},m}$ less than $s(n-\tau)$. □

Notice for the $w_{t,j}$ weights of Proposition 3.57 that 1 has been added to all of them only to ensure that they are all non-negative.

Proposition 3.58. *In the context of Proposition 3.57, the worst-case complexity of finding a satisfactory Q as a minimal $\Phi_{1,\boldsymbol{w}}$-weighted element in the row space of $A^{\mathcal{H}}_{s,\ell}$ is as in Table 3.4, for various choices of module minimisation algorithm.*

Proof. The proposition follows from Table 2.4 on page 42 after estimating the measures of $\Phi_{1,\boldsymbol{w}}(A^{\mathcal{H}}_{s,\ell})$; a rough estimate will suffice for us.

Clearly we can overestimate the max-degree by

$$\begin{aligned}
\operatorname{maxdeg}(\Phi_{1,\boldsymbol{w}}(A^{\mathcal{H}}_{s,\ell})) &\le \operatorname{maxdeg}_{t,h}(\Pi(H^{(t)}_{[h]})) + \operatorname{maxdeg}\Xi + \max_{t,h}\{w_{t,h}\} \\
&\le s\deg_x R + (q+1) + (2 + q^{-1}(q+1)(q-1) + q^{-1}\ell m)
\end{aligned}$$

Note by Theorem 3.52, Point 2 that $\ell m < s(n-\tau)$. Now $\deg_x R \le q^{-1}\deg_{\mathcal{H}} R$, and assuming that we constructed R using Lemma 3.50 on page 81 then $\deg_{\mathcal{H}} R < n+2g$. Therefore the above is in $O(sq^{-1}n + sq)$. This is $O(sq^{-1}n)$ assuming $q \in O(q^{-1}n)$.

For the orthogonality defect, we just employ the straightforward bound

$$\Delta(\Phi_{1,\boldsymbol{w}}(A^{\mathcal{H}}_{s,\ell})) \le q(\ell+1)\operatorname{maxdeg}(\Phi_{1,\boldsymbol{w}}(A^{\mathcal{H}}_{s,\ell})) \in O(\ell s n)$$ □

The estimate $O(n^{5/3}\ell^3 s\log(n)^{O(1)})$ by using the GJV is, when focusing on dependence on n, the fastest known technique for decoding Hermitian codes, and has apparently not been obtained in the literature before now. See also the discussion in Section 3.6

Remark. It is quite possible that one can slightly improve the asymptotic bounds by a more careful analysis of the orthogonality defect, akin to the estimate for GRS codes, where we obtained dependence on $(n-k)$ instead of n. ◆

Complexity for computing Q for Guruswami–Sudan decoding Hermitian codes

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$q\ell^3 sn^2$	$n^{7/3}\ell^3 s$
Alekhnovich	$\mathrm{M}(q\ell)\mathrm{P}(\ell sn)\log(n) + q^2\ell^2\mathrm{P}(sq^{-1}n)$	$n^2\ell^4 s\log(n)^{2+o(1)}$
GJV	$\mathrm{M}(q\ell)\mathrm{P}(sq^{-1}n)\log(q\ell sn)^{O(1)}$	$n^{5/3}\ell^3 s\log(n)^{O(1)}$

Table 3.4: In all the estimates, we have assumed $q \in O(q^{-1}n)$, i.e. that at least around q^2 affine points were chosen. For relaxation, we have also assumed $n \approx q^3$, i.e. that more or less *all* affine points were chosen, and for logs we have used $s, \ell \in n^{O(1)}$.

Remark. One can easily show that using $\nu = q$ and $w_{t,j} = (q+1)j + tm$ instead of what we use in Proposition 3.57 will yield $\deg \Phi_{q,\boldsymbol{w}}(\Upsilon_z(Q)) = \deg_{\mathcal{H},m} Q$ for any $\Upsilon_z(Q)$ in the row space of $A^{\mathcal{H}}_{s,\ell}$. Clearly then the minimal $\deg_{\mathcal{H},m}$-weighted element in Q is found as a *minimal* weighted-degree vector in the row space of $A^{\mathcal{H}}_{s,\ell}$.

This is a more "true" weighing than what we have used, where we can only ensure that $\deg_{\mathcal{H},m} Q$ will be *low enough* if $\Upsilon(Q)$ is the minimal weighted-degree vector. Since the Mulders–Storjohann and Alekhnovich algorithms do not get a performance penalty for $\nu > 1$, these weights yield exactly the same performance. This is exactly what Brander proposed in [Bra10] using the Alekhnovich algorithm, and it is very related to what Lee and O'Sullivan propose [LO09] when using their Mulders–Storjohann variant. Both of them give slightly worse complexity estimates than that of Table 3.4, though; this is possibly due to our use of the orthogonality defect which improves the complexity analysis. See also Section 3.6. For $\nu > 1$, however, the GJV *does* take a performance penalty of roughly a factor ν; so with these "truer" weights, its performance would be $O(n^2\ell^3 s\log(n)^{O(1)})$. ◆

Remark. It is natural and interesting to compare these complexities with those of decoding GRS codes, e.g from Table 3.1 on page 63 or Table 3.3 on page 76; evidently, decoding Hermitian codes is slower, but Hermitian codes do allow much longer codes relative to the field size. Unfortunately, I have not had time to implement this method and prepare an example to compare with GRS codes. It should be mentioned that these speeds for decoding Hermitian codes match or beat the best previously known decoding methods, as discussed in Section 3.6. ◆

3.6 Related work

Bounds on list decoding

The upper bound on decoding radius for Guruwami–Sudan, the Johnson radius $J(n,d) = n - \sqrt{n(n-d)}$, is an intriguing one. Using the original Johnson bound [Joh62] on the size of constant weight codes, one can show that for *any* linear code of length n and minimum distance d, any Hamming ball of radius $J(n,d) - \varepsilon$, $\varepsilon > 0$, the

number of codewords in this ball is upper bounded by $O(1/\varepsilon)$. I.e. it is a "constant" in terms of n, which means that the Johnson radius is an a priori lower bound on the possible radius one can list decode any code in polynomial time.

The converse also holds true, in the following sense. It was shown by Goldreich, Rubinfeld and Sudan [GRS00] that there exists infinite families of codes containing arbitrary long codes, each with relative minimum distance δ, such that there in each code is a ball of radius greater than $J(n, n\delta)$ containing $\Omega(2^n)$ codewords. In other words: a polynomial time list decoder decoding further than $J(n, n\delta)$ is impossible for this family of codes, since it would have to produce an exponentially long list of results. We do not know whether Reed–Solomon codes are such codes, and no polynomial time algorithm decoding asymptotically beyond the Johnson radius for these codes is known. List decoding bounds are a deep combinatoric field, and we will not discuss more on this; a good introduction, complete reference and all but very recent results can be found in Guruswami's book [Gur07].

Confining ourselves to codes over $\mathbb{F}_q$, there is the q-ary Johnson radius: $J_q(n, d) = \frac{q-1}{q}(n - \sqrt{n(n - \frac{q}{q-1}d)})$. Though perhaps not immediately obvious, $J_q(n, d) > J(n, d)$ for all legal values of the parameters. In particular for $q = 2$, the difference is quite big. Confined to q-ary codes and families of such, exactly the same as the above two statements hold. A variant of the Guruswami–Sudan can actually decode Alternant codes, being sub-field sub-codes of GRS codes, up to $J_q(n, d)$; more on this a bit further down.

The Johnson bound [Joh62] gives a maximal number of codewords in a ball of any radius less than the Johnson radius. For $q \to \infty$, this bound was slightly improved by Cassuto and Bruck [CB04], and the list size bound for the Guruswami–Sudan algorithm of Corollary 3.14 is exactly this. However, that does not mean that the Guruswami–Sudan algorithm will ever return that many codewords; in random experiments, it is for most code parameters exceedingly rare that it even returns more than a single codeword. There are, however, certain evidence that the number of codewords will match the list size for certain (large) parameters and specially constructed received words; in particular in [JH01], but see also the extensive discussion in Guruswami's thesis [Gur01, Section 4.8, but also elsewhere].

As mentioned, I have been unable to find good and closed expressions for the choice of the parameters s and ℓ in the literature. The analysis of Section 3.1.2 was originally developed for the Wu algorithm for a preprint version of [BHNW13], but recast for this thesis for the Guruswami–Sudan (see duality in Section 5.2.2). Independently of us, Trifonov and Lee [TL12] did a similar analysis of the parameter choices in Wu's algorithm, which one could probably apply to the Guruswami-Sudan as well, and one might very well reach the same expressions as ours. I have not seen the relaxation of the list size in Corollary 3.13 on page 55, nor the asymptotic analysis elsewhere.

Other interpolation methods

As already remarked, the interpolation method of Section 3.2 follows closely the one of Lee and O'Sullivan [LO08]; in fact, the only difference is how we obtain the shortest vector from the row space of $A_{s,\ell}$ from (3.3) on page 61. As discussed toward the end of Section 2.7, their algorithm is a reinvention of the Mulders–Storjohann algorithm, but due to an inferior complexity analysis, they report the complexity $O(\ell^4 s n^2)$. Alekhnovich proposed an interpolation method using his own minimisation algorithm [Ale05], but his explicit basis was much larger than that of $A_{s,\ell}$, resulting in a worse complexity (though still quasi-linear in n). Beelen and Brander [BB10] were the first to combine Lee and O'Sullivan's module approach with the Alekhnovich minimisation algorithm. They used a slightly worse basis than $A_{s,\ell}$ which they then rewrote using some of the same techniques as we used when deriving the 2D key equation in Section 3.4; the result is a basis with the same max-degree as that of $A_{s,\ell}$ (they do remark toward the end that a reviewer made them aware of the possibility of using $A_{s,\ell}$ instead). They end up reaching the same speed as in Table 3.1 on page 63 with the Alekhnovich algorithm.

Cohn and Heninger [CH10] generalised work by Coppersmith [Cop97], Howgrave-Graham [HG01], and May [May03] for solving certain polynomial equations over the integers, to the case of function fields, by which one arrives at exactly the Guruswami–Sudan algorithm. For Reed–Solomon codes, the explicit basis of $A_{s,\ell}$ is also quickly apparent. They propose to find the shortest vector of the row space using the GJV, and so naturally arrives at the same complexity as ours (it should be noted that their notation is very different from ours, however, making direct comparison cumbersome). Independently, Bernstein [Ber11a] also pointed out the same equivalence between the Guruswami–Sudan algorithm and the above-mentioned work of Coppersmith and Howgrave-Graham. Bernstein also proposed to use the GJV for module minimisation.

Trifonov proposed a different way to utilise Gröbner bases over modules for the interpolation problem [Tri10b]: one can combine Gröbner bases for small s and ℓ into bases for larger ones by element-wise multiplication of the bases followed by a Gröbner basis computation; by "binary exponentiation", one arrives at the desired s and ℓ after logarithmically many such combinations. He relies on fast multiplication techniques but still estimates a complexity of $O\big(s^3 n(n-\sqrt{n}+\log(n))\big)$, which is worse than the module minimisation approach using D&C techniques, see Table 3.1 on page 63. Furthermore, his analysis is involved and relies on an algebraic assumption he states is known to be false. There is also room for improvement: Trifonov uses Lee–O'Sullivan's module minimisation for the intermediate Gröbner basis computations, i.e. basically Mulders–Storjohann, and one can readily replace these by the Alekhnovich algorithm or the GJV, which should speed up computations (Trifonov himself suggests the former in his article on the Wu list decoder [Tri10a], but has, to my knowledge, not examined the performance of doing so). Also, it would be of prime interest to see if one can utilise the orthogonality defect measure to

prove that the intermediate Gröbner basis computations are indeed fast if using the Mulders–Storjohann or Alekhnovich algorithm; just as we did for the Multi-trial algorithm of Section 3.3.

The Kötter interpolation method is yet another Gröbner basis approach for constructing an interpolation polynomial [Köt96], which was generalised to handle the Guruswami–Sudan case by Nielsen and Høholdt [NH98]. No running time was reported in the latter, but by inspecting the pseudo-code it seems to be $O(s^3\ell^2 n^2)$, thus comparable to the method of Section 3.2 using Mulders–Storjohann, but slower than the D&C-approaches, see Table 3.1 on page 63.

As mentioned previously, the 2D key equation formulation of finding a satisfactory Q was first given by Zeh, Gentner and Augot [ZGA11]. They also gave an extended version of the Berlekamp–Massey algorithm to solve the equation, and for this application, they report a complexity of $O(\ell s^4 n^2)$. This is faster than ours using Mulders–Storjohann or the Demand–Driven algorithm since $s < \ell$. Two things might be noted here: first, they assume that $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau) \approx 0$ in an asymptotic sense, such that they can replace $\ell s(n-\tau) - \binom{\ell+1}{2}(k-1)$ with $\binom{s+1}{2}n$; it is possible that by going carefully through the analysis of the Mulders–Storjohann or Demand–Driven algorithm for this special case, one can utilise that assumption to similar effect. In any case, one should here be wary of the warnings of Section 1.2. Second, as described in Section 2.7, there is hope that using a better order of row-reductions in the Demand–Driven algorithm and improving analysis might yield another reduction since the algorithms are fundamentally similar. On the other hand, I know of no D&C variant of the algorithm of [ZGA11] which might solve the 2D key equation in a complexity with quasi-linear dependence on n, such as we accomplish using the Alekhnovich algorithm or the GJV. The above discussion also applies to the case $s = 1$ where the Roth–Ruckenstein algorithm is faster than our non-D&C solutions; see also Section 2.7.

Guruswami–Sudan list decoding other codes

Kötter and Vardy showed in their famous paper [KV03a] how Guruswami–Sudan can be used for *soft-decision decoding* of GRS codes, by using different multiplicities for each possible received symbol: if we are fairly confident of a symbol on a certain position, we give it high multiplicity, while if we are uncertain, we give it low to none. By rigorously analysing the choices of multiplicities, they showed how the soft-decision information could be put to efficient use. They suggested finding the interpolation polynomial using the Kötter method (see above). Lee and O'Sullivan [LO06] showed how to compute an initial basis analogous to $A_{s,\ell}$ in (3.3) on page 61, though they did not write it up explicitly. One could then apply any of the module minimisation methods, and Lee and O'Sullivan apply their Mulders–Storjohann variant and analyse its complexity; I am not aware of anyone having looked into using other module minimisation algorithms.

Since Alternant codes are sub-field sub-codes of GRS codes, the Guruswami–Sudan algorithm can also decode these up to $J(n, d)$, where d is the designed minimum distance, i.e. the minimum distance of the enclosing GRS code. Using a clever multiplicity assignment, inspired by their soft-decision variant of Guruswami–Sudan, Kötter and Vardy showed how to decode up to the q-ary Johnson radius $J_q(n, d)$ (see above) where q is the size of the small field that the Alternant code is in. Kötter and Vardy circulated a preprint which is now not easily found, but never published their results; the method is however well described by Roth [Rot06] and by Augot, Barbier and Couvreur [ABC10]. Bernstein got to the same result [Ber11a] using the approach of Coppersmith and Howgrave-Graham already mentioned (citing in particular a paper by Coppersmith, Howgrave-Graham and Nagaraj [CHGN08]), and described how to find the interpolation polynomial using the GJV; his reported complexity is $O(\ell \mathrm{M}(\ell) n \log(\ell n)^{O(1)})$. We get back to this decoding method for $q = 2$ by a relation to Wu list decoding binary Goppa codes in Section 5.3.2.

For AG codes, Brander in his thesis [Bra10] gives a concise description of how to use a similar module minimisation approach for the class of AG codes constructed with simple C_{ab} curves, of which the Hermitian curves are members. As already remarked, the weights he applies for the minimisation problem have $\nu = q$; when using the Alekhnovich algorithm this is not a problem, but since we wanted to also apply the GJV, we wanted to avoid the performance penalty of ν. Like for the case of GRS codes, Brander used a slightly inferior explicit basis than $A^{\mathcal{H}}_{s,\ell}$, so he reached the complexity $O\big(n^2\ell^5 \log^{2+o(1)}(n)\big)$ for decoding an Hermitian code; using the basis $A^{\mathcal{H}}_{s,\ell}$ replaces a factor ℓ with a factor s. As mentioned, Lee and O'Sullivan also presented essentially the same technique but using an extension of their Mulders–Storjohann variant for module minimisation. They arrive at the complexity $O\big(n^{8/3}\ell^3 s^2\big)$, which is for some reason worse than what we get using Mulders–Storjohann; I have no explanation for this discrepancy. Brander's thesis contains a comparison with other strategies for finding Q for AG codes [Bra10, Section 4.2.2], all of which are slower than his method, and therefore ours. The technique of Lee and O'Sullivan has since been generalised to other AG codes in various publications, e.g. [GMR12], and for soft-decision decoding [LO10], and it would be interesting to see if for these generalised works, one can also readily replace the module minimisation techniques with the faster D&C algorithms. Cohn and Heninger [CH10] can decode any AG code using their Coppersmith–Howgrave-Graham language of Guruswami–Sudan, but since they do not give a way to put the resulting minimisation problem into an $\mathbb{F}[x]$-module form, they can give no better complexity than "polynomial time" through Gaussian elimination.

Apart from the generalisation of Guruswami–Sudan to AG codes, the algorithm was furthermore conceptually generalised to the class of "Ideal codes" by Guruswami, Sahai and Sudan [GSS00], though this conceptual algorithm only runs in polynomial time assuming certain basic algorithms exist for the codes in question; one of these is a polynomial-time method to construct a Q polynomial. The class of Ideal codes include the Chinese Remainder codes, for which the LLL algorithm [LLL82] is used

for constructing Q; apparently, the lattice in which they find a Q polynomial as a short vector is rather large, and it would be interesting if one could use techniques like those of Section 3.2 for searching in a smaller lattice.

The interpolation ideas of the Guruswami–Sudan algorithm also form the basis of the decoding algorithms for the various "folded" codes, which can be list decoded arbitrarily close to capacity $1 - \frac{k}{n}$ (for very large alphabets, and some only in probabilistic polynomial time), see e.g. [GR06, GX12, GW12].

Similar to the 2D key equations of Section 3.4, Beelen and Høholdt obtained Hankel-form matrix equations for certain AG codes using linear algebraic rewriting [BH08b]. It is interesting whether it is immediately or easily possible to solve these 2D key equations fast using the module minimisation methods presented in Chapter 2.

Other multi-trial approaches

Not much work has been done similar to the Multi-Trial algorithm of Section 3.3. Cassuto, Bruck and McEliece [CBM13] described an iterative Guruswami–Sudan, where one chooses final s, ℓ, but from the beginning only utilise a subset of the usually allowed $x^i y^j$-monomials for constructing $Q(x, y)$. One then finds a Q which can correct fewer errors, but it is also cheaper to construct; if no nearby codewords are found, one adds monomials and constructs a better Q. The spirit is therefore similar to ours. However the computational saving is far from ours, with only a factor 2 saved when comparing the complexity of correcting a single error with correcting up to the decoding radius; furthermore, the method is based on the Kötter interpolation method.

Tang, Chen and Ma [TCM12] gave a different iterative variant based on this interpolation method, and applied to the Kötter–Vardy soft-decision variant of Guruswami–Sudan [KV03a]. There is little analysis on the complexity, however, making direct comparison with ours (in the hard-decision case) difficult; being based on Kötter's interpolation method, it is clear it must have at least quadratic dependence on n. It is not clear whether our methods of Section 3.3 generalise to the soft-decision case: we seem to be crucially dependent on the nice form of the explicit basis $A_{s,\ell}$, leading to the recurrence relations of Lemma 3.30 and Lemma 3.32.

One could also employ Trifonov's binary exponentiation interpolation method (see above) in a multi-trial manner. After each intermediate combination and Gröbner basis reduction, one indeed has a Gröbner basis for intermediate values of s and ℓ. However, due to the nature of the binary exponentiation, it would be non-trivial to let these intermediate parameter values achieve useful values, i.e. yielding high intermediate decoding radii, while at the same time resulting in only logarithmically many combination steps.

Chapter 4

Power decoding

Power decoding, or "virtual extension to an interleaved code", is a more recent paradigm for decoding low-rate codes beyond half the minimum distance than the Guruswami–Sudan. It was originally developed by Schmidt, Sidorenko and Bossert for GRS code [SSB06]. Their description stems from the surprising fact that a received word coming from a low-rate GRS code can be "powered" to give a received word of a higher-rate GRS code having the same error positions. This additional received word usually aids in identifying the error positions, making one capable of correcting more errors. In rare cases, the new restrictions add no new information and the method then fails. The method is not a list decoder since it either returns exactly one result or it fails; in particular it fails whenever there are more than one codeword equally close to the received word.

A classical way to decode GRS up to half the minimum distance is to solve the so-called "Key Equation", that is, a simple key equation where the main unknown is the error locator: a polynomial identifying the erroneous positions by its roots. In Power decoding, each "power" of the received word has its own key equation, so one ends up with a 2D key equation in the error locator with $\rho > 1$ and $\sigma = 1$. Schmidt and Sidorenko proposed to solve this using an extension of the Berlekamp–Massey algorithm; see also the discussion in Section 2.7. Not surprisingly, we will solve the 2D key equation using module minimisation techniques.

We will begin in Section 4.1, not with the original description of Schmidt, Sidorenko and Bossert, but by deriving an alternative formulation which we call "Power Gao decoding", where we instead "power" the key equation which is at the heart of the Gao decoding algorithm [Gao03]. This new formulation is very easy to derive and

its behaviour is arguably easier to analyse than the original. However, in Section 4.2 we derive the original "Power Syndromes" formulation from Power Gao, and show that the two are in fact equivalent. In Section 4.3 we expound on certain deep connections between Power decoding and with Sudan decoding, i.e. the Guruswami–Sudan algorithm with $s = 1$. In Section 4.4 we extend Power Gao to Hermitian codes, and again solve the emerging 2D key equation using the results of Chapter 2.

Contributions

- The Power Gao formulation, as well as the Proposition 4.10 on page 103 describing the equivalence between Power Gao and Power Syndromes.
- Proposition 4.7 on page 101 which proves that the failure behaviour of Power decoding is invariant under shifts by codewords.
- Applying module minimisation methods to Power decoding. Using the GJV or Alekhnovich algorithm, this is faster than previously known methods for this decoding paradigm.
- Proposition 4.13 on page 106 giving an orthogonality between Power Syndromes and the Roth–Ruckenstein method for finding Q in Sudan decoding.
- Theorem 4.17 on page 111 upper bounding the failure probability of Power decoding to that of Coppersmith–Sudan decoding. Also the remark on page 112 that the same matrix has Power decoding solutions in its left kernel and Sudan decoding solutions in its right kernel.
- The Power Gao formulation for Hermitian codes.
- Applying module minimisation techniques for Power decoding Hermitian codes, resulting in the fastest known algorithms for this approach.
- A rigorous and precise analysis of the decoding radius of Power decoding Hermitian codes.

4.1 Empowering Gao

We will now show how the polynomials that we have already dealt with in the last chapter, R and G, turn out to be involved in a different key equation relation than that of Section 3.4, and how this can be used for a different key equation-based decoding algorithm; naturally, the resulting key equation can be solved using the methods of Chapter 2. To begin, we need to introduce a number of new polynomials; Proposition 4.2 relating them will seem like magic. Indeed, the unfair amount of foresight in the definitions of the in-going polynomials is based on a long history of key equation decoding.

Recall that $\boldsymbol{c} = \mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f)$ was the sent word, achieved as the evaluation of some information polynomial f, that $\boldsymbol{r}$ is the received word and that we let $\boldsymbol{r}' = (\frac{r_1}{\beta_1}, \ldots, \frac{r_n}{\beta_n})$. Recall also $R(x)$ and $G(x)$ of Definition 3.22 on page 60. For the former, we gener-

alise this to the *power Lagrangian* of the received word:

$$R^{(t)}(x): \quad R^{(t)}(\alpha_i) = r_i'^t, \quad \text{for } i = 1, \dots, n \tag{4.1}$$

Just as for $R(x)$, we have $\deg R^{(t)}(x) \leq n-1$ and $R^{(t)}$ can be easily calculated by the receiver. We will also introduce a *power error vector* $\boldsymbol{e}^{(t)} = (e_1^{(t)}, \dots, e_n^{(t)})$ where $e_i^{(t)} = (r_i^t - c_i^t)/\beta_i^t = r_i'^t - f(\alpha_i)^t$. Finally, introduce $\zeta_i = \prod_{j \neq i}(\alpha_i - \alpha_j)^{-1}$ for each $i = 1, \dots, n$; note that $R^{(t)} = \sum_{i=1}^n r_i'^t \zeta_i \prod_{j \neq i}(x - \alpha_j)$.

Definition 4.1. The error-locator and the power error-evaluator is defined as

$$\Lambda(x) = \prod_{j \in \mathcal{E}}(x - \alpha_j)$$
$$\Omega^{(t)}(x) = \sum_{j \in \mathcal{E}} \zeta_j e_j^{(t)} \frac{\Lambda(x)}{x - \alpha_j} = \sum_{j \in \mathcal{E}} \zeta_j e_j^{(t)} \prod_{h \in \mathcal{E} \setminus \{j\}} (x - \alpha_h)$$

Note that $\deg \Omega^{(t)} < \deg \Lambda = |\mathcal{E}|$. Clearly, if we can find Λ and $\Omega^{(1)}$ then we can immediately decode: Λ will tell us the error positions as its roots, and $\Omega^{(1)}$ will evalute to the error value multiplied with a known constant for each of those error positions. Once Λ and $\Omega^{(1)}$ have been obtained, a faster, but essentially equivalent, strategy is *Forney's method*, see e.g. [Bos99, p. 71].

Remark. The reader familiar with decoding methods using the error-locator and error-evaluator might be surprised at our definition; for example, the error-locator is often defined as $\overline{\Lambda}(x) = \prod_{j \in \mathcal{E}}(1 - x\alpha_j)$, thus revealing the inverse of the errors positions' evaluation points (recalling the $\overline{p}(x)$-notation, see Appendix A). We call this the "indirect error-locator" It stems from the classical key equation using the classical syndromes, and we will meet it in Section 4.2; since we will begin with the Gao key equation, however, we will use the "direct error-locator" now. ◆

Choose now some $\ell \in \mathbb{N}$ subject to $\ell(k-1) < n$. Then we have the following fundamental decoding relation which allows us to find Λ and $\Omega^{(t)}$ in a variety of ways:

Proposition 4.2. *For $t = 1, \dots, \ell$ then*

$$\Lambda R^{(t)} - \Omega^{(t)} G = \Lambda f^t$$

Proof. The above is equivalent to $R^{(t)} - \Omega^{(t)}\Upsilon = f^t$, where $\Upsilon = \prod_{i \notin \mathcal{E}}(x - \alpha_i)$ is the *truth-locator*. We will show that the two sides of this equation evaluate to the same at all n different α_i; this will imply that they are equivalent modulo G but since $\deg R^{(t)} < n$, $\deg(\Omega^{(t)}\Upsilon) < |\mathcal{E}| + (n - |\mathcal{E}|) = n$ and $t \deg f < n$ that means they are equal.

First, for $i \notin \mathcal{E}$ then immediately $R^{(t)}(\alpha_i) = f^t(\alpha_i)$ and $\Upsilon(\alpha_i) = 0$. For $i \in \mathcal{E}$, note that $\Omega^{(t)}(\alpha_i) = \zeta_i e_i^{(t)} \prod_{h \in \mathcal{E} \setminus \{i\}}(\alpha_i - \alpha_h)$, since all but the ith term in the sum

vanishes. Therefore

$$R^{(t)}(\alpha_i) - \Omega^{(t)}(\alpha_i)\Upsilon(\alpha_i) = (f(\alpha_i)^t + e_i^{(t)}) - \zeta_i e_i^{(t)} \prod_{h \in \mathcal{E}\setminus\{i\}} (\alpha_i - \alpha_h) \prod_{j \notin \mathcal{E}} (\alpha_i - \alpha_j)$$

which reduces to $f(\alpha_i)^t$. □

The first way which we are going to decode using Proposition 4.2 is by the following "Power Gao" key equation:

Corollary 4.3. *For $t = 1, \ldots, \ell$ we have*

$$\Lambda R^{(t)} \equiv \Lambda f^t \mod G(x) \tag{4.2}$$

Therefore $(\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$ is a solution to the 2D key equation of Type 2 having parameters

- $\rho = 1$, $\sigma = \ell$.
- $G_j^{\mathrm{KE}} = G(x)$ *for* $j = 1, \ldots, \ell$.
- $S_{1,j}^{\mathrm{KE}} = R^{(j)}(x)$, *for* $j = 1, \ldots, \ell$.
- $\nu = 1$ *and* $\eta_1 = \ell(k-1) + 1$.[1]
- $w_j = (\ell - j)(k-1)$ *for* $j = 1, \ldots, \ell$.

Proof. Equation (4.2) follows immediately from Proposition 4.2, which means that most of the 2D key equation parameters follow directly from mapping to Problem 2.31 on page 26; only η_1 and the w_j need a brief explanation: for a 2D key equation of Type 2, we should set those such that $\deg \Lambda + \eta_1 > \max_j\{\deg(\Lambda f^j) + w_j\}$. Any setting of η_1 and the w_j satisfying this would mean that $(\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$ is a solution, but in order to disallow as many other solutions as possible, we should maximise w_j under these constraints. The chosen weights are then chosen minimally under these considerations (adding some number $\delta \in \mathbb{N}$ to all η_i and w_j will not change the set of solutions). □

The matrix to be minimised when solving the 2D key equation of Proposition 4.2 is a weighted version of:

$$M_{\mathrm{Gao}} = \left(\begin{array}{c|cccc} 1 & R^{(1)} & R^{(2)} & \cdots & R^{(\ell)} \\ \hline & G & & & \\ & & G & & \\ & & & \ddots & \\ & & & & G \end{array}\right)$$

We then have a decoding algorithm: solve for the minimal solution of the 2D key equation of Corollary 4.3. If this solution has the very special structure of $\boldsymbol{s} = (\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$, then return this. For added confidence, one could verify that $\deg \Lambda = \mathrm{dist}(c - \mathrm{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f)) \leq \tau$ for the chosen decoding radius τ.

[1] In [Nie13b, p. 881], it was erroneously stated that $\eta_1 = \ell(k-1)$.

However, we have not stated that $\boldsymbol{s}$ is a *minimal* solution to the 2D key equation so we are not sure that $\boldsymbol{s}$ will be found by using the methods of Chapter 2. Even stronger, we actually need it to be the *only* minimal solution, up to $\mathbb{F}$-scalar multiples, for otherwise it depends on rather arbitrary computations during the chosen module minimisation whether $\boldsymbol{s}$ will emerge as the computed solution or not. Let us therefore say that decoding using Power Gao *fails* if $\boldsymbol{s}$ is *not* the only minimal solution to the 2D key equation (up to $\mathbb{F}$-scalar multiples).

It turns out that for only few errors beyond half the minimum distance, and assuming random error patterns, the probability that we fail is very small; not surprisingly, the probability then increases with the number of errors and at a certain point, we can be certain we will fail. Simulations indicate that the increase in failure probability is extremely slow up until very close to this upper bound on the radius, and then it goes rapidly to 1. We have not found an analytical upper bound for the failure probability and this is still an open problem; we will discuss it again twice, both in Section 4.2 and Section 4.3.2, directly relating it to failure probabilities of other methods (whose failure probability have not been determined well either). We can, however, find the ends of the interval by inspecting the 2D key equation:

Proposition 4.4. *In the context of Corollary 4.3, whenever* $|\mathcal{E}| \leq \frac{n-k}{2}$ *then the vector* $(\Lambda, \Lambda f, \ldots, \Lambda f^{\ell})$ *is the minimal solution to the 2D key equation up to* $\mathbb{F}$ *scaling. It is* not *a minimal solution whenever*

$$|\mathcal{E}| > \tfrac{\hat{\ell}}{\hat{\ell}+1} n - \tfrac{1}{2}\hat{\ell}(k-1) - \tfrac{\hat{\ell}}{\hat{\ell}+1} \tag{4.3}$$

where $\hat{\ell} = \min\left\{\ell,\ \left\lfloor \sqrt{(\frac{1}{2}+\frac{1}{k-1})^2 + \frac{2(n-2)}{k-1}} - (\frac{1}{2}+\frac{1}{k-1}) \right\rfloor\right\}$, *except possibly under special circumstances pertaining to a certain system of equations specified in the proof.*

Proof. Consider first the lower bound: let (λ, ψ) be a minimal solution to the 2D key equation for $\ell = 1$ while $|\mathcal{E}| \leq \frac{n-k}{2}$, and we will show that $\lambda = \Lambda$ and $\psi = \Lambda f$. Thus by assumption $\deg\lambda \leq \deg\Lambda$ and $\deg\lambda + k > \deg\psi$. By Proposition 4.2, we have $R = f + \Omega^{(1)}\Upsilon$. Since (λ, ψ) satisfies the key equation, G divides $\lambda R - \psi = \lambda(f + \Omega^{(1)}\Upsilon) - \psi = \lambda f - \psi + \lambda\Omega^{(1)}\Upsilon$, and since $\Upsilon \mid G$ that means $\Upsilon \mid (\lambda f - \psi)$. Now $\deg(\lambda f - \psi) \leq \deg\lambda + k - 1$, so if not $\lambda f - \psi = 0$ then we get $\deg\Upsilon = n - |\mathcal{E}| \leq \deg\lambda + k - 1$. But $\deg\lambda \leq \deg\Lambda$ so that immediately yields the contradiction $2|\mathcal{E}| \geq n - k + 1$. Therefore $\lambda f = \psi$, so plugging this back in the key equation we get $G \mid (\lambda R - \lambda f)$ which implies $\Lambda \mid \lambda$; obviously, the minimal such solution must be $\lambda = \Lambda$.

For the upper bound, let $\boldsymbol{s} = (s_0, s_1, \ldots, s_\ell)$ be a minimal solution to the 2D key equation. Since $\boldsymbol{s}$ is a solution and the key equation is of Type 2, then $\deg s_0 + \eta_1 > \max_{j=1,\ldots,\ell}\{\deg s_j + w_j\}$ which means $\deg(\Phi_{1,\bar{\boldsymbol{w}}}(\boldsymbol{s})) = \deg s_0 + \eta_1$. We will prove that $\deg(\Phi_{1,\bar{\boldsymbol{w}}}(\boldsymbol{s})) \leq \lceil \frac{\hat{\ell}}{\hat{\ell}+1} n + (\ell - \frac{1}{2}\hat{\ell})(k-1) + \frac{1}{\hat{\ell}+1} \rceil$, where $\hat{\ell}$ is as the proposition states, and that whenever the expression within $\lceil\cdot\rceil$ is not integer, then there is another minimal solution which differs on the first $1 + \hat{\ell}$ positions by more than an $\mathbb{F}$ scaling.

Since $\eta_1 = \ell(k-1)+1$ this means that when $\deg\Lambda > \frac{\hat{\ell}}{\hat{\ell}+1}n - \frac{1}{2}\hat{\ell}(k-1) - \frac{\hat{\ell}}{\hat{\ell}+1}$, then $(\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$ can't be a minimal solution. When $\deg\Lambda$ *equals* this right-hand side, on the other hand, our sought vector is a minimal solution, and it might well be the only one (up to $\mathbb{F}$ scaling).

To prove the claim on d, we will employ Corollary 2.40 on page 34. Reusing notation there, we have $\zeta_j = n + (\ell - j)(k-1)$ for $j = 1, \ldots, \ell$ and $\zeta_{\ell+1} = -\infty$, as well as

$$\begin{aligned} d_j &= (j+1)^{-1}\Big(\ell(k-1) + 1 + \textstyle\sum_{h=1}^{j}\big((\ell-h)(k-1)+n\big)\Big) = \ldots \\ &= \tfrac{j}{j+1}n + \tfrac{1}{j+1} + (\ell - \tfrac{1}{2}j)(k-1) \end{aligned}$$

where M_j is the first $j+1$ rows and columns of M_{Gao}. The corollary states that for the greatest integer $0 < j \leq \ell$ such that $\zeta_{j+1} \leq \lceil d_j \rceil < \zeta_j$, then $\deg(\Phi_{1,\vec{\boldsymbol{w}}}(\boldsymbol{s})) \leq \lceil d_j \rceil$. Since ζ_j is an integer for all j, this is the same as satisfying $\zeta_{j+1} \leq d_j + 1 < \zeta_j$. Assume first $j < \ell$. Then after some rearranging, the requiring amounts to

$$\sqrt{(\tfrac{1}{2} + \tfrac{1}{k-1})^2 + \tfrac{2(n-1)}{k-1}} - (\tfrac{3}{2} + \tfrac{1}{k-1}) < j \leq \sqrt{(\tfrac{1}{2} + \tfrac{1}{k-1})^2 + \tfrac{2(n-2)}{k-1}} - (\tfrac{1}{2} + \tfrac{1}{k-1})$$

Now, if the lower bound is at least ℓ, then the assumption must have been false, so $j = \ell$: the upper bound is then surely satisfied which implies $d_\ell < \zeta_\ell$ and we of course have $d_\ell \geq \zeta_{\ell+1} = -\infty$, so indeed $j = \ell$ is a valid, and obviously maximal, choice.

If the lower bound is less than ℓ, however, we see that the allowed interval for j has width at most 1. So it contains at most one integer; however, it *must* contain one integer, by Corollary 2.40. This integer is therefore the floor of the right-hand side, and that, single allowable choice of j, must be maximal.

The resulting choice of j is exactly $\hat{\ell}$; Therefore, by Corollary 2.40, $\deg(\Phi_{1,\vec{\boldsymbol{w}}}(\boldsymbol{s})) \leq \lceil d_{\hat{\ell}} \rceil$ and if $\lceil d_{\hat{\ell}} \rceil$ is not an integer, there are multiple $\mathbb{F}$-linearly independent minimal solutions; the corollary states that this is true, except possibly if a certain system of equations one can get from the 2D key equation has a specific property described in Proposition 2.39 on page 33. □

Remark. In spite of the lack of finality in the statement of Proposition 4.4, it should be emphasised how precisely it seems to predict what actually happens in simulations. When $|\mathcal{E}|$ is less than or equal to the right-hand side of (4.3), we almost always decode correctly. For the case of equality, this means that almost always, even though the sought vector $(\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$ has *exactly* the expected degree of a minimal solution to the 2D key equation, this is the only minimal solution (up to $\mathbb{F}$ scaling). When $|\mathcal{E}|$ is greater than the given bound, we almost always fail. However, and I believe this observation is new, it turns out that we *do* succeed once in a while. For instance, on a $[10, 2, 9]$ GRS code with $\ell = 3$, Proposition 4.4 predicts that we should be able to decode at most 5 errors; in a simulation with 20 000 random error patterns of weight 6, however, we decoded successfully in 0.02% of the cases. ◆

Remark. Compare the decoding radius upper bound of Proposition 4.4 with that of Sudan decoding on Proposition 3.18 on page 58 assuming $\ell = \hat{\ell}$: their difference

is only $\frac{\ell}{\ell+1}$, and even less when considering the bound here is $\leq$ while that of Sudan is $<$. For many parameter choices, when relaxing to integers they become the same, but occasionally Sudan decoding can correct one more error. ♦

Remark. Proposition 4.4 clearly implies that picking ℓ so high that $\hat{\ell} < \ell$ does not make sense: it won't improve the decoding radius. Therefore, the maximal sensible choice of ℓ is $\left\lfloor \sqrt{(\frac{1}{2}+\frac{1}{k-1})^2 + \frac{2(n-2)}{k-1}} - (\frac{1}{2}+\frac{1}{k-1}) \right\rfloor$. Compare this with the choice of ℓ from Proposition 3.18 on page 58: for high n and k they are almost the same. It is not too difficult to find examples where they differ by 1; however, I have been unable to find such an example which also produced a decoding radius which was off by more than the single error remarked upon above. ♦

Example 4.5. *Consider again the [250, 40, 211] code from Example 3.20 on page 59. We get* $\hat{\ell} = \min\{\ell,\ 3.08\}$. *Choosing* $\ell = 3$, *then the right-hand side of (4.3) gives* $128\frac{1}{4}$. *Thus, we can hope of decoding success only when* $|\mathcal{E}| \leq 128$*; note that this is one less than for Sudan decoding.*

The parameters of the 2D key equation of Corollary 4.3 become

$$\eta_1 = 118 \qquad \operatorname{maxdeg}\left(\Phi_{1,\bar{\boldsymbol{w}}}(M_{\text{Gao}})\right) = n + w_1 = 328$$
$$w_j = [78, 39, 0] \qquad \Delta(\Phi_{1,\bar{\boldsymbol{w}}}(M_{\text{Gao}})) = n + w_1 - 1 = 327$$

Compare this with Example 3.43 on page 77.

In a trial of 10 000 decodings of random error patterns of weight 128, success was in 99.99%. *Performing the same number of trials with errors of weight 129 failed every time.* ♠

This decoder is not a *list* decoder, since it returns at most one codeword, and it is therefore important to be clear on how the decoder behaves in the presence of more than one codeword within the upper bound on decoding radius. To each of these codewords will correspond an information word and error locator, and these will be solutions to the 2D key equation. Since the minimal of these solutions will be the one with the error locator of lowest degree, then *when* the decoder returns a codeword it will always be the *closest* codeword. If the two codewords have different distance to the received, then the decoder might succeed in finding the closest, or it might fail altogether. If two codewords are equally close to the received, however, then the decoder must fail (by our definition of "fail"): they will both be minimal solutions whence the minimal solution is non-unique. Lastly, it should be remarked that the decoder might also fail even when there is only one codeword within the decoding distance, simply due to an unintended small solution to the 2D key equation. All of these cases have indeed been observed in experiments using Codinglib [Nie13a].

Remark. We call this decoding method "Power Gao" since it is a power-extended version of what one could call the Gao key equation: $\Lambda R \equiv \Lambda f \mod G$. In his paper on decoding GRS codes [Gao03], Gao never wrote it as such, but it is exactly this relation he is using to decode by running the Euclidean algorithm on R and G and

halting when the remainder has degree less than $(n+k)/2$. Choosing $\ell = 1$ and solving the 2D key equation of Corollary 4.3 using Mulders–Storjohann is, step-wise computationally, exactly the same as this. Mulders–Storjohann would compute a matrix $\Phi_{1,\bar{\boldsymbol{w}}}(\left[\begin{smallmatrix}\boldsymbol{g}_1\\ \boldsymbol{g}_2\end{smallmatrix}\right]) \in \mathbb{F}[x]^{2\times 2}$ in weighted weak Popov form, and this is also just what the Euclidean algorithm would compute: in particular $g_{1,2}$ and $g_{2,2}$ are the "remainders" on and before the iteration where the Euclidean algorithm should be terminated in Gao decoding, since $g_{i,1}R + h_iG = g_{i,2}$ for $i = 1, 2$ for some $h_i \in \mathbb{F}[x]$ not recorded in our matrix representation. Note also how the decoding limits of Proposition 4.4 coincide at this case of $\ell = 1$, as was expected. ◆

Remark. If Power Gao decoding fails, then we know by Theorem 2.35 that $\boldsymbol{s}$ can be constructed as an $\mathbb{F}[x]$-linear combination of the rows of the found $\Phi_{1,\bar{\boldsymbol{w}}}$-weighted weak Popov form; furthermore, we have upper bounds on the degrees of the coefficient polynomials, assuming that $\deg \Lambda < \tau$ for some chosen τ. One could therefore exhaustively search through this space, looking for the very special structure of $\boldsymbol{s}$. Of course, the complexity of this is exponential in the difference between τ and the degrees of the rows of the matrix in $\Phi_{1,\bar{\boldsymbol{w}}}$-weighted weak Popov form (see Theorem 2.35), but if these are all 0 or small, one might consider doing it.

Wu's decoding algorithm, which Chapter 5 is devoted to, is a special case of this strategy for $\ell = 1$, i.e. classical key equation solving, where one uses a method closely related to the polynomial interpolation of Guruswami–Sudan to "search" through this exponential space in polynomial time. ◆

Proposition 4.6. *The worst-case complexity of finding Λ and f as a minimal solution to the 2D key equation of Corollary 4.3, or fail, is as in Table 4.1, for various choices of module minimisation algorithm.*

Proof. Follows from Table 2.4 on page 42 after estimating the measures of the 2D key equation. Remember that $\ell(k-1) < n$, from which it easily follows that

$$\gamma = \operatorname{maxdeg}(\Phi_{1,\bar{\boldsymbol{w}}}(M)) = \max\{\eta_1, \deg G_1^{\mathrm{KE}} + w_1, \ldots, \deg G_\rho^{\mathrm{KE}} + w_\rho\} \in O(n)$$
$$\delta = \Delta(\Phi_{1,\bar{\boldsymbol{w}}}(M)) < \rho\gamma \ \in O(n)$$ □

Remark. The complexity-dominating part of the work for Power Gao decoding is performed when searching for Λ and f; the remaining work can be done in $O(n \log n)$. Similarly for the methods in Chapter 3, the dominating part is in finding a satisfactory $Q(x, y)$; thus, we can fairly compare the complexities of Table 4.1 with those of Table 3.1 on page 63; Table 3.2 on page 69; and Table 3.3 on page 76 when setting $s = 1$, to determine which decoding method is fastest. Note how Power Gao decoding provides a factor 1 to ℓ speedup when using D&C algorithms, while it gives a factor ℓ^2 speedup when comparing the D–D algorithm of Table 4.1 with other non-D&C methods. As theorised in Section 2.7, it might be possible to improve the Demand–Driven algorithm when applied to the Q-finding key equations to reduce or remove this gap. In particular, the algorithm of Roth and Ruckenstein to solve the

Complexity of finding Λ with Power Gao key equation of Corollary 4.3

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^2 n^2$	$\ell^2 n^2$
Alekhnovich	$\mathrm{M}(\ell)\mathrm{P}(n)\log(n) + \ell^2\mathrm{P}(n)$	$\ell^3 n \log(n)^{2+o(1)}$
GJV	$\mathrm{M}(\ell)\mathrm{P}(n)\log(\ell n)^{O(1)}$	$\ell^3 n \log(n)^{O(1)}$
D–D	$\ell n P(n)$	$\ell n^2 \log^{1+o(1)}(n)$

Table 4.1: For relaxation, we have used the same rules as in Table 3.1 on page 63. Remember that $\tilde{P}(n)$ for the D–D algorithm is $\mathrm{P}(n) = n \log n \log\log n$ since $G_j^{\mathrm{KE}}(x)$ are not powers of x.

Sudan key equations (see Section 3.4.1) runs in $O(\ell n^2)$ and so matches our Power Gao decoding complexity (ignoring the log-factors of the latter). ◆

It is a fundamentally aesthetic property of any algebraic decoder that it behaves invariant under which codeword was sent, so its behaviour depends only on the error vector. For analysis, it can also be an important simplifying step. Since for $t > 1$ then $e^{(t)}$ depends on the sent codeword, one could be concerned that the Power Gao decoder does not have this property; fortunately, we have:

Proposition 4.7. *Power Gao decoding fails for some received word $\boldsymbol{r}$ if and only if it fails for $\boldsymbol{r} + \hat{\boldsymbol{c}}$ where $\hat{\boldsymbol{c}}$ is any codeword.*

Proof. We will show that Power Gao decoding fails for $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ if and only if it fails for $\boldsymbol{e}$ as received word; since $\boldsymbol{c}$ was arbitrary, that implies the proposition.

Let $R_e^{(t)}(x)$ be the power Lagrangians for $\boldsymbol{e}$ as received word, i.e. $R_e^{(t)}(\alpha_i) = (e_i/\beta_i)^t$ for each i and t and let $R_e = R_e^{(1)}$. Consider a solution to the corresponding 2D key equation $(\lambda, \psi_1, \ldots, \psi_\ell)$; i.e. $\lambda R_e^{(t)} \equiv \psi_t \mod G$ and $\deg\lambda + t(k-1) + 1 > \deg\psi_t$. Let as usual $R^{(t)}(x)$ be the power Lagrangians for $\boldsymbol{r}$ as received word and $R = R^{(1)}$. Note now that $R^{(t)} \equiv R^t \mod G$ since both sides of the congruence evaluate to the same at all α_i; similarly $R_e^{(t)} \equiv R_e^t \mod G$. Since $r_i' = f(\alpha_i) + e_i/\beta_i$ we have by linearity of Lagrangian interpolation that $R(x) = f(x) + R_e(x)$. Define $\psi_0 = \lambda$ and note that then also for $t = 0$ we have $\deg\lambda + t(k-1) + 1 > \deg\psi_t$. We then have the congruence chain modulo G:

$$\lambda R^{(t)} \equiv \lambda R^t \equiv \lambda(f + R_e)^t \equiv \lambda \sum_{s=0}^{t} \tbinom{t}{s} f^s R_e^{t-s} \equiv \sum_{s=0}^{t} \tbinom{t}{s} f^s \psi_{t-s} \mod G$$

Each term in the last sum has degree $s\deg f + \deg\psi_{t-s} < s(k-1) + \deg\lambda + (t-s)(k-1) + 1 = \deg\lambda + t(k-1) + 1$, which means that

$$\left(\lambda,\ \sum_{s=0}^{1} \tbinom{1}{s} f^s \psi_{1-s},\ \ldots\ ,\ \sum_{s=0}^{\ell} \tbinom{\ell}{s} f^s \psi_{\ell-s}\right)$$

is a solution to the 2D key equation with $\boldsymbol{r}$ as a received word. The same argument holds in the other direction, so any solution to one of the key equations induces a

solution to the other with the same first component; obviously then, their minimal solutions must be in bijection, which directly implies that they either both fail or neither fail. □

4.2 Empowering the classical syndromes

Now we will derive the original Power decoding as proposed by Schmidt, Sidorenko and Bossert [SSB06,SSB10]. Just as they did, we will now assume that $\alpha_i \neq 0$ for all i which is equivalent to $x \nmid G$. Recall the "reversal" notation $\overline{p}(x)$, see Appendix A, and introduce then a set of power series, strongly reminiscent of (3.9) on page 72:

$$S_\bullet^{(t)}(x) = \frac{\overline{R}^{(t)}(x)}{\overline{G}(x)} \tag{4.4}$$

In Proposition 4.11, we specify the exact relation between these two power series. For now, using the results of the last section, we immediately get:

Corollary 4.8. *For $t = 1, \ldots, \ell$ then*

$$\overline{\Lambda} S_\bullet^{(t)} \equiv \overline{\Omega}^{(t)} \mod x^{n-t(k-1)-1} \tag{4.5}$$

Therefore $(\overline{\Lambda}, \overline{\Omega}^{(1)}, \ldots, \overline{\Omega}^{(\ell)})$ is a solution to the 2D key equation of Type 2 having parameters

- $\rho = 1$, $\sigma = \ell$.
- $G_j^{\mathrm{KE}} = x^{n-j(k-1)-1}$ *for* $j = 1, \ldots, \ell$.
- $S_{1,j}^{\mathrm{KE}} = (S_\bullet^{(j)}(x) \bmod x^{n-j(k-1)-1})$, *for* $j = 1, \ldots, \ell$.
- $\nu = 1$, $\eta_1 = 0$ *and* $w_j = 0$ *for* $j = 1, \ldots, \ell$.

Proof. Reversing both sides of the equation of Proposition 4.2, we get

$$\overline{\Lambda}\,\overline{R}^{(t)} - \overline{\Omega}^{(t)}\overline{G} = \overline{\Lambda}\;\overline{f}^{\,t} x^{n-t(k-1)-1}$$

Reducing this modulo $x^{n-t(k-1)-1}$ and dividing by $\overline{G}$ on both sides gives (4.5); note that we can divide by $\overline{G} = {}^{[n]}\overline{\prod_{i=1}^{n}(x-\alpha_i)}$ since no $\alpha_i = 0$ and therefore $\gcd(x, \overline{G}) = 1$.

For the modelling as a 2D key equation, we proceed exactly as for Corollary 4.3 excepting that we reduce $S_\bullet^{(t)}(x)$ modulo $G_j^{\mathrm{KE}} = x^{n-j(k-1)-1}$ first. □

The matrix to be minimised when solving the 2D key equation of Proposition 4.2 is a weighted version of:

$$M_{\mathrm{Syn}} = \left(\begin{array}{c|cccc} 1 & S^{(1)} & S^{(2)} & \cdots & S^{(\ell)} \\ \hline & x^{n-k} & & & \\ & & x^{n-2k+1} & & \\ & & & \ddots & \\ & & & & x^{n-\ell(k-1)-1} \end{array}\right)$$

It is fascinating to compare this with the matrix for finding Q in Guruswami–Sudan in the case $s = 1$, (3.12) on page 77. This duality, as well as the fact that they are the same for $\ell = 1$, is expounded in Section 4.3.1.

Example 4.9. *For the [250, 40, 211] code of Example 4.5 on page 99 and $\ell = 3$, the parameters of the 2D key equation of Corollary 4.8 become*

$$\operatorname{maxdeg}\left(\Phi_{1,\bar{\boldsymbol{w}}}(M_{\mathrm{Gao}})\right) = n = 250 \qquad \Delta(\Phi_{1,\bar{\boldsymbol{w}}}(M_{\mathrm{Gao}})) = n - 1 = 249$$

Compare this with Example 3.43 on page 77 and Example 4.5 on page 99. ♠

Obviously, this "Power Syndromes" key equation leads to a decoding algorithm just as for the Power Gao key equation. We could proceed as before and deduce an upper bound on the decoding radius, but the following result is much stronger and more interesting.

Proposition 4.10. *Decoding using Power Gao fails if and only if decoding using Power Syndromes fails.*

Proof. Power Gao fails if there is some $\lambda \in \mathbb{F}[x]$ which is not a constant times the sought Λ and such that $\deg\lambda \leq \deg\Lambda$ and $\psi^{(t)}(x) = (\lambda R^{(t)} \bmod G)$ has $\deg\lambda + \ell(k-1) + 1 > \deg\psi^{(t)} + (\ell - t)(k-1)$ i.e. $\deg\psi^{(t)} < \deg\lambda + t(k-1) + 1$ for each $t = 1, \ldots, \ell$. This means there must be some $\omega^{(t)}(x)$ with $\deg\omega^{(t)} \leq \deg\lambda - 1$ such that

$$\begin{aligned} \lambda R^{(t)} - \omega^{(t)} G &= \psi && \iff \\ \overline{\lambda}\,\overline{R}^{(t)} - \overline{\omega}^{(t)}\overline{G} &= \overline{\psi}^{(t)} x^{\deg G + \deg\lambda - 1 - (\deg\lambda + t(k-1))} && \implies \\ \overline{\lambda}\,\overline{R}^{(t)} &\equiv \overline{\omega}^{(t)}\overline{G} \mod x^{n-t(k-1)-1} \end{aligned}$$

Dividing by $\overline{G}$, we see that $\overline{\lambda}$ and the $\overline{\omega}^{(t)}$ satisfy the congruence equations as well as the degree bounds which are necessary to form a solution to the Power Syndrome key equation. Showing the proposition in the other direction runs analogously. □

Clearly, the Power Syndromes decoding algorithm then has the same upper bound on its decoding radius as well as the same failure probability as Power Gao decoding. Schmidt, Sidorenko and Bossert did not compute this failure probability in general, but they did upper bound it for $\ell = 2$ when the codes are over binary extension fields [SSB10, Theorem 3]: they reach an upper bound of $\frac{(\gamma q^3)^{|\mathcal{E}|}}{q^{3\tau_{\max}}(q-1)}$, where γ is a specific number slightly greater than 1, and $\tau_{\max}$ is the right-hand side of (4.3) on page 97. This was generalised to general q in [ZWB12], still only for $\ell = 2$. This is quickly decaying exponentially for $|\mathcal{E}|$ moving away from the upper bound. We discuss failure probability again in Section 4.3.2.

The utilised coefficients of the power series $S_\bullet^{(1)}$ are usually arrived at by starting with a parity check matrix for the GRS code; for indeed, it is exactly the classical syndrome polynomial, used for decoding GRS codes for decades. It is also simply a shifted version of $S_\circ^{(1,1)}$ of (3.9) on page 72, which we used for the Guruswami–Sudan

Q-finding key equations in Section 3.4. The following proposition sums up these relations for all $t \geq 1$:

Proposition 4.11. *For $t = 1, \dots, \ell$, there exists $U^{(t)} \in \mathbb{F}[x]$ with $\deg U^{(t)} < (t-1)(n-1)$ and such that $S_\circ^{(t,1)}(x) = U^{(t)}(x) + x^{(t-1)(n-1)} S_\bullet^{(t)}(x)$. Furthermore,*

$$S_\bullet^{(t)}(x) \equiv \sum_{i=1}^{n} \frac{e_i^{(t)} \zeta_i}{1 - x\alpha_i} \equiv \sum_{j=0}^{n-k-1} x^j \sum_{i=1}^{n} e_i^{(t)} \zeta_i \alpha_i^j \mod x^{n-t(k-1)-1}$$

and is the classical syndrome polynomial for the received word $(r_1'^t \beta_1, \dots, r_n'^t \beta_n)$ in the GRS code $[n, t(k-1)+1, n-t(k-1)]$ having $\boldsymbol{\alpha}, \boldsymbol{\beta}$ as evaluation points and column multipliers respectively.

Proof. Note first that $R^t \equiv R^{(t)} \mod G$ since $R^t(\alpha_i) = R^{(t)}(\alpha_i) = r_i'^t$ for each i. This means that for $t > 1$ there exists $q \in \mathbb{F}[x]$ with $\deg q \leq t(n-1) - n$ such that

$$R^t - qG = R^{(t)} \qquad \iff \qquad \overline{R}^t - \overline{q}\overline{G} = x^{(t-1)(n-1)} \overline{R}^{(t)}$$

where the exact power of x after the reversal of coefficients comes from $\deg R^t = t(n-1) > n$ when $t > 1$. Dividing the reversed equation by $\overline{G}$ gives the sought, where $\overline{q} = U$ and since then $\deg U \leq t(n-1) - n < (t-1)(n-1)$ as promised.

For the syndrome characterisation, we simply insert the definition of the power error-evaluator into (4.5):

$$\overline{\Lambda} S_\bullet^{(t)} \equiv \sum_{j \in \mathcal{E}} \zeta_j e_j^{(t)} \frac{\overline{\Lambda}(x)}{1 - x\alpha_j} \mod x^{n-t(k-1)-1}$$

We can divide through by $\overline{\Lambda}$ since $\gcd(x, \overline{\Lambda}) = 1$ since 0 is not an evaluation point, and the first equivalence follows. The second equivalence comes simply from expanding the fraction $\frac{1}{1-x\alpha_i}$ to a power series. For $S_\bullet^{(1)}(x)$, this is clearly the classical syndrome polynomial, see e.g. [Rot06, (6.3) on p. 185][2]. For $t > 1$, it follows by inserting the modified code parameters. □

Remark. The shift of $(t-1)(n-1)$ from $S_\circ^{(t,1)}$ to $S_\bullet^{(t)}(x)$ equals v_t in Proposition 3.38 on page 72 when $s = 1$; so $S^{(t,1)}(x)$ of that proposition equals $S_\bullet^{(t)}(x) \bmod x^{m_0 - u_t}$. Together with the additional Roth–Ruckenstein reduction in Section 3.4.1, we end up using exactly the same part of $S_\bullet^{(t)}(x)$ in the two methods. ◆

Remark. The usual way of deriving the Power Syndrome equations of Corollary 4.8, as introduced by Schmidt, Sidorenko and Bossert in [SSB06], begins by remarking exactly that from one received word $(r_1, \dots, r_n)$, one can "virtually extend" to ℓ received words in progressively larger codes in the manner described by the above proposition. One can consider the *interleaved* code from these ℓ GRS codes, and the ℓ "virtual words" as one codeword in this code. Finding the common error locator

[2] In that section of [Rot06], GRS codes are introduced using parity check matrices, and one needs also [Rot06, Problem 1 on p. 211] to see that our ζ_i are equivalent to his v_i.

Complexity of finding Λ by solving Power Syndrome key equation of Corollary 4.8

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^2(n-k)^2$	$\ell^2 n^2$
Alekhnovich	$\mathrm{M}(\ell)\mathrm{P}(n-k)\log(n-k) + \ell^2\mathrm{P}(n-k)$	$\ell^3 n \log(n)^{2+o(1)}$
GJV	$\mathrm{M}(\ell)\mathrm{P}(n-k)\log(\ell(n-k))^{O(1)}$	$\ell^3 n \log(n)^{O(1)}$
D–D	$\ell(n-k)^2$	ℓn^2

Table 4.2: For relaxation, we have used the same rules as in Table 3.1 on page 63. Remember that $\tilde{P}(n)$ for the D–D algorithm is $O(n)$ since all $G_j^{\mathrm{KE}}(x)$ are powers of x. One should keep in mind that the decoder works best for low rates, so $n-k$ is close to n.

using multiple key equations for general interleaved GRS codes had already been examined by these authors in [SS06]. This description originally gave the name "decoding by virtually extending to an interleaved Reed–Solomon code" to Power decoding. ♦

Proposition 4.12. *The worst-case complexity of finding for $\overline{\Lambda}$ and the $\overline{\Omega}^{(t)}$ as a minimal solution to the 2D key equation of Corollary 4.8, or fail, is as in Table 4.2, for various choices of module minimisation algorithm.*

Proof. Follows from Table 2.4 on page 42 after estimating the measures of the 2D key equation. Since all weights are trivial $\Phi_{1,\vec{\boldsymbol{w}}}(M) = M$, where M is the relevant matrix to minimise. It easily follows that both $\gamma = \mathrm{maxdeg}\,(M)$ and $\delta = \Delta(M)$ are in $O(n-k)$. □

Remark. From classical coding theory, "syndromes" are usually reserved for values derived from the received word which depend only on the error vector and not the codeword. For $S^{(1)}(x)$, this is true for the first $n-k$ coefficients, by the above proposition. However for $S^{(t)}(x), t > 1$, then $e^{(t)}$ itself depends on the sent codeword. This immediately raises the question whether the failure behaviour of Power decoding is invariant under shifting by codewords; we saw in Proposition 4.7 that this was indeed so for Power Gao decoding, so by Proposition 4.10, it must also hold for Power Syndrome decoding. ♦

4.3 Connections with Sudan decoding

Clearly, Power decoding and Guruswami–Sudan for $s = 1$, i.e. Sudan decoding, must be closely related: as already remarked, they decode up to nearly the same radius, and the utilised part of the power series $S_\bullet^{(t)}(x)$ occurs also in the Roth–Ruckenstein speedup Section 3.4.1.

In this section we give two new algebraic connections between Power decoding and Sudan decoding, demonstrating that there is an even more intimate relation between them; in a sense, we will show how they are "dual" to each other.

The connection of Section 4.3.1 seems to be completely of theoretical interest, while the connection in Section 4.3.2 might help analyse the failure probability of Power decoding. Though I have not succeeded in doing so, it is my hope that both of the connections can help in finding a "Power decoding with multiplicities" algorithm, assuming that such a method should be dual to the Guruswami–Sudan algorithm in the way that Power decoding is dual to Sudan.

4.3.1 Orthogonality with Roth–Ruckenstein

For half-the-minimum distance decoding, there is a well-known connection between the error-locator from the key equations and the Guruswami–Sudan algorithm with $s = \ell = 1$.[3] Specifically, let $Q(x, y)$ be a minimal interpolation polynomial for Guruswami–Sudan with $s = \ell = 1$, see Theorem 3.2 on page 48. Since $Q(\alpha_i, r'_i) = 0$ for all i as well as $0 = Q(x, f(x)) = Q_0(x) + f(x)Q_1(x)$, then clearly $Q_0 = -Q_1 f$, which means $Q_1(\alpha_i)f(\alpha_i) + r'_i Q_1(\alpha_i) = 0$ for all i. Whenever $i \in \mathcal{E}$ this means $Q_1(\alpha_i) = 0$, which implies that $\Lambda \mid Q_1$. However, we are decoding up to half the minimum distance which means $\tau < n - \tau$, and therefore $\hat{Q} = \Lambda f + y\Lambda$ is an interpolation polynomial satisfying the degrees; by the minimality of Q with the requirement that $\Lambda \mid Q_1$, we must have $Q = \gamma\hat{Q}$ for some $\gamma \in \mathbb{F}$.

Note that in Power Gao for $\ell = 1$, i.e. regular Gao decoding, we would also search for $(\Lambda, \Lambda f)$; indeed, if we compare the matrix used for Q-finding of Guruswami–Sudan by the method of Section 3.2, i.e. $A_{1,1}$ from page 61, with the one for Power Gao for $\ell = 1$, we see that they are the same except for the order of the columns!

Assume again $\alpha_i \neq 0$. The following is a generalisation of the above connection for $\ell > 1$ and $s = 1$, which ties together all the important variables of Power Syndrome decoding with those of Guruswami–Sudan key equation interpolation:

Proposition 4.13. *Let $Q(x, y) = \sum_{t=0}^{\ell} Q_t(x)y^t$ be a valid interpolation polynomial for the Guruswami–Sudan algorithm with $s = 1$ and some ℓ. Let $\tilde{B}(x) = \tilde{B}_0(x)$ with $\tilde{B}_0(x)$ as in (3.11) on page 77. Let $\Lambda, \Omega^{(1)}, \ldots \Omega^{(\ell)}$ be the error-locator and the power error-evaluators respectively. Then*

$$\left(\overline{\Lambda},\ \overline{\Omega}^{(1)},\ x^{k-1}\overline{\Omega}^{(2)},\ \ldots\ ,\ x^{(\ell-1)(k-1)}\overline{\Omega}^{(\ell)}\right) \cdot \left(-\tilde{B},\ \overline{Q}_1,\ \overline{Q}_2,\ \ldots\ ,\ \overline{Q}_\ell\right) = 0$$

Proof. As remarked on page 104, $S^{(t,1)}(x) \equiv S^{(t)}_{\bullet}(x) \mod x^{n-t(k-1)-1}$ when $s = 1$, where $S^{(t,1)}$ is as in Proposition 3.38 on page 72. Thus, from (4.5) we have

$$\begin{aligned} \overline{\Lambda}(x)S^{(t)}_{\bullet}(x) &\equiv \overline{\Omega}^{(t)}(x) && \mod x^{n-t(k-1)-1} \\ \overline{\Lambda}(x)S^{(t,1)}(x)x^{(t-1)(k-1)} &\equiv x^{(t-1)(k-1)}\overline{\Omega}^{(t)}(x) && \mod x^{n-k} \end{aligned} \quad \Longleftrightarrow$$

[3] The latter is occasionally called the Welch–Berlekamp algorithm, after the patented decoding algorithm in [WB86]. This decoder has little to do with the Guruswami–Sudan for $s = \ell = 1$ on the surface, but algebraic connections between the two decoders have led them to be considered "the same". To me, it seems that the algebraic connections between Gao decoding and Guruswami–Sudan for $s = \ell = 1$ are tighter, however.

Beginning from (3.11) on page 77 and inserting the above, we get:

$$\begin{aligned}
\sum_{t=1}^{\ell} \overline{Q}_t(x) S^{(t,1)}(x) x^{(t-1)(k-1)} &\equiv \tilde{B}(x) && \bmod x^{n-k} && \iff \\
\sum_{t=1}^{\ell} \overline{Q}_t(x) \overline{\Lambda}(x) S^{(t,1)}(x) x^{(t-1)(k-1)} &\equiv \overline{\Lambda}(x) \tilde{B}(x) && \bmod x^{n-k} && \iff \\
\sum_{t=1}^{\ell} \overline{Q}_t(x) x^{(t-1)(k-1)} \overline{\Omega}^{(t)}(x) &\equiv \overline{\Lambda}(x) \tilde{B}(x) && \bmod x^{n-k}
\end{aligned}$$

Recall $\nabla_t^Q = s(n-\tau) - t(k-1)$ and $\deg Q_t < \nabla_t^Q$. Therefore, the left-hand side has degree at most $\max_t\{(\nabla_t^Q - 1) + (t-1)(k-1) + (\tau - 1)\} = n - k - 1$, and the right-hand side has degree at most $\tau + (n - k - \tau - 1) = n - k - 1$. Thus the two sides both have degree below the modulus and must be equal. □

For $\ell = 1$, the above is equivalent to the connection mentioned initially, i.e. $\Lambda \mid Q_1$, since

$$\left(\overline{\Lambda},\ \overline{\Omega}^{(1)}\right) \cdot \left(-\tilde{B},\ \overline{Q}_1\right) = 0 \qquad \iff \qquad \overline{\Lambda}\tilde{B} = \overline{Q}_1 \overline{\Omega}^{(1)}$$

Since by their definitions $\overline{\Lambda}$ and $\overline{\Omega}^{(1)}$ share no roots, they are coprime, so the above implies $\overline{\Lambda} \mid \overline{Q}_1$ which is the same as $\Lambda \mid Q_1$.

Remark. The above "duality" of the two types of key equations, (3.11) and (4.5), is well-known for Padé approximations: as indicated on Table 2.1 on page 28 and further discussed in Section 2.7, (3.11) has the form, but not type, of a Hermite Padé approximation, while (4.5) has the form, but not type, of a Simultaneous Padé approximation. The duality of Hermite and Simultaneous Padé approximations, in exactly the sense of Proposition 4.13, is described in e.g. [BGM96].

It has also been generalised for so-called Matrix-Padé approximations [BL94], see also Table 2.1. An analogous duality could certainly be given for two 2D key equations which are similarly strongly connected in the $S_{i,j}$ and the G_j. If there exists a form of Power decoding involving multiplicities, I imagine that the resulting 2D key equation would be dual to the Q-finding 2D key equations of Section 3.4 in this manner. ◆

Remark. For $s = 1$, from (3.7) on page 71 one immediately gets $\sum_{t=1}^{\ell} Q_t(x) R(x)^t \equiv -Q_0 \mod G$, and combining this key equation with the Power Gao key equation (4.2) on page 96, one can get a duality exactly like above. Interestingly, this ends up simply stating $\sum_{t=0}^{\ell} Q_t(x) f(x)^t = 0$, i.e. that $f(x)$ is a y-root of $Q(x, y)$; which we of course already knew. Since Power Gao and Power Syndromes are connected via the simple rewrite across the fundamental relation of Proposition 4.2 on page 95, it would seem that the relation of Proposition 4.13 is just an incarnation of the same fact. ◆

4.3.2 Across Coppersmith–Sudan

The Coppersmith–Sudan algorithm [CS03] is an alternative list-decoder, decoding errors almost up to the Guruswami–Sudan radius[4]. This algorithm is very simple and has virtues that the Guruswami–Sudan does not, e.g. that it immediately generalises to decode Interleaved Reed–Solomon codes. It is also clearly connected to the Guruswami–Sudan decoder: as we will see, and as the authors themselves remarked, a central matrix appears in both Coppersmith–Sudan and the simplest way of calculating an interpolation polynomial in Guruswami–Sudan.

For $s = 1$, we will give some new connections between Coppersmith–Sudan and Power Syndrome decoding; in fact, Power Syndrome decoding can be seen as a Fourier transformed analogue of Coppersmith–Sudan. This will transitively reveal yet another facet of the relation between Power Syndrome and Roth–Ruckenstein interpolation.

First, we describe briefly the Coppersmith–Sudan algorithm for the special case of $s = 1$. For notational brevity, we need to introduce the following short-hands, for $\boldsymbol{v} \in \mathbb{F}^b$ and $a, b, t \in \mathbb{N}$:

$$V_a(\boldsymbol{v}) = \begin{pmatrix} v_1^0 & v_1^1 & \dots & v_1^{a-1} \\ \vdots & & \ddots & \vdots \\ v_b^0 & v_b^1 & \dots & v_b^{a-1} \end{pmatrix} \qquad \boldsymbol{v}^t = (v_1^t, \dots, v_b^t)$$

Let τ be a decoding radius to be upper bounded shortly, and define $\grave{\nabla}_t^Q = n - \tau - t(k-1) - 1 = \nabla_t^Q - 1$. Define now the component matrices C_t:

$$C_t = \begin{pmatrix} \alpha_1^0 r_1'^t & \alpha_1^1 r_1'^t & \dots & \alpha_1^{\grave{\nabla}_t^Q - 1} r_1'^t \\ & \vdots & & \\ \alpha_n^0 r_n'^t & \alpha_n^1 r_n'^t & \dots & \alpha_n^{\grave{\nabla}_t^Q - 1} r_n'^t \end{pmatrix} = \operatorname{diag}(\boldsymbol{r}'^t) \cdot V_{\grave{\nabla}_t^Q}(\boldsymbol{\alpha})$$

The total matrix C is then $C = [C_0 \mid C_1 \mid \dots \mid C_\ell]$ and has dimensions $n \times \grave{\nabla}_\Sigma^Q$ where $\grave{\nabla}_\Sigma^Q = \sum_{t=0}^{\ell} \grave{\nabla}_t^Q$.

Consider now the sub-matrix $\grave{C}$ consisting of the rows of C corresponding to positions not in error. Any column of $\grave{C}$ can be seen as the evaluations of some monomial $x^a y^t$ at $(\alpha_i, f(\alpha_i)), i \notin \mathcal{E}$, i.e. evaluating $x^a (f(x))^t$ at α_i. Since $a + t(k-1) < \grave{\nabla}_t^Q + t(k-1) = n - \tau - 1$, then $x^a (f(x))^t$ is a polynomial of degree at most $n - \tau - 2$. The fundamental observation that Coppersmith and Sudan make is that $\grave{C}$ has rank at most $n - \tau - 1$: any column can be written as a linear combination of the columns $1, \dots, n - \tau - 2$, since these correspond to evaluating the polynomials $1, x, \dots, x^{n-\tau-2}$. C therefore has rank at most $n - \tau - 1 + |\mathcal{E}|$. If there are at least

[4] In [CS03], they report the same decoding radius, but as we will see, they made a small but important mistake which decrease the decoding radius slightly

n columns, i.e. $\grave{\nabla}^Q_\Sigma \geq n$, and if $|\mathcal{E}| \leq \tau$, then $n - \tau - 1 + |\mathcal{E}|$ is less than the generic rank n, which can be exploited to identify the non-error rows constituting $\grave{C}$. The requirement on the number of columns leads to having to choose:[5]

$$\tau < \tfrac{\ell}{\ell+1} n - \tfrac{1}{2}\ell(k-1) - \tfrac{\ell}{\ell+1} \tag{4.6}$$

Just like for Power decoding. From this intuition, Coppersmith and Sudan prove the following three observations:

Proposition 4.14 (Coppersmith–Sudan)**.**

1. *If $|\mathcal{E}| \leq \tau$ then C has a non-trivial left kernel [CS03, Claim 2].*
2. *Any non-zero element in the left kernel will have at least $n - |\mathcal{E}|$ non-zero positions [CS03, Claim 3].*
3. *Assuming that $\boldsymbol{e}$ is picked uniformly at random from all vectors in $\mathbb{F}^n$ of Hamming weight $|\mathcal{E}| \leq \tau$, then for any non-zero vector $\boldsymbol{v}$ in this left kernel, then $\operatorname{supp}(\boldsymbol{v})$ will be disjoint with $\mathcal{E}$ with probability at least $(1 - \frac{\ell}{q-1})^{|\mathcal{E}|}$, where q is the cardinality of the field, assuming $\ell < q$ [CS03, Lemma 4 slightly modified].*

The decoding algorithm is then clear: construct C and find any vector $\boldsymbol{v}$ in the left kernel space of C. $\boldsymbol{v}$ probably has at least $n - \tau$ non-zero positions, all of which are probably non-errors; erase all the other positions and perform erasure-decoding.

By the third point in the above, this algorithm can fail if the found vector in the left kernel of C is *not* non-zero in only the non-error positions. Similar to how we defined "failure" for Power decoding, let us consider Coppersmith–Sudan to fail if there is any such vector in the left kernel of C. We will show that whenever Power Syndrome decoding fails, exactly this occurs.

Remark. What is extremely fascinating is that in the *right* kernel of C lies the Q-polynomials of Sudan decoding! Interpreting the requirements of Theorem 3.2 on page 48 for an interpolation polynomial $Q(x, y) = \sum_{i=0}^{\ell} Q_t(x) y^t$ as a system of linear equations in the coefficients of Q, then the system matrix is almost identical to C: the difference is that in Sudan decoding, we allow $\deg_{1,k-1}(Q) \leq n - \tau - 1$ so $\deg Q_t \leq \grave{\nabla}^Q_t$. Therefore, the sub-matrices corresponding to C_t will have $\grave{\nabla}^Q_t + 1$ columns, while C_t has only $\grave{\nabla}^Q_t$ columns. In Sudan decoding, searching instead a Q such that $\deg_{1,k-1} Q < n - \tau - 1$ could be done as finding a vector in the right kernel of C; the decoding radius of the Sudan algorithm would then decrease to exactly that of (4.6). ◆

Now for the connection to Power decoding: we will show that any element in the left kernel of C gives rise to a solution to the Power Syndrome key equation of

[5] Note that there is an important mistake in [CS03]: In Section 2.1 and 2.2 of that paper, $\mathcal{M}_T$ is defined as all monomials in x and y of $(1, k-1)$-weighted degree *at most* T; however, from the proofs in Section 2.3 (in particular for Claim 2, line 6 and Claim 3 line 3) it is clear that they assume $\mathcal{M}_T$ as all monomials of weighted degree *less than* T. This off-by-one corresponds exactly to our use of $\grave{\nabla}^Q_t$ instead of ∇^Q_t, which is why we reach the slightly reduced decoding radius.

Corollary 4.8 on page 102. Recall that $\zeta_i = \prod_{j \neq i}(x - \alpha_j)^{-1}$ and let $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_n)$. A well-known identity for Vandermonde matrices is that for any $m < n$:

$$(\operatorname{diag}(\boldsymbol{\zeta})V_m(\boldsymbol{\alpha}))^\top V_{n-m}(\boldsymbol{\alpha}) = \mathbf{0}$$

We have therefore that $\operatorname{diag}(\boldsymbol{\zeta})V_{\tau+1}(\boldsymbol{\alpha})C_0 = \mathbf{0}$, so by multiplying the entire C on the left with $\operatorname{diag}(\boldsymbol{\zeta})V_{\tau+1}(\boldsymbol{\alpha})$ we get:

$$\begin{aligned} \operatorname{diag}(\boldsymbol{\zeta})V_{\tau+1}(\boldsymbol{\alpha})C &= [\mathbf{0} \mid Z_1 \mid \ldots \mid Z_\ell] && \text{where} \\ Z_t &= \big(\operatorname{diag}(\boldsymbol{\zeta})V_{\tau+1}(\boldsymbol{\alpha})\big)^\top \operatorname{diag}(\boldsymbol{r}'^t)V_{\vec{\nabla}_t^Q}(\boldsymbol{\alpha}) \\ &= \left[\sum_{i=1}^{n} \alpha_i^{a+b-2} r_i'^t \zeta_i\right]_{(a,b)=(1,1),\ldots,(\tau+1,\vec{\nabla}_t^Q)} \end{aligned}$$

Note how Z_t has Hankel form. Define $Z = [Z_1 \mid \ldots \mid Z_\ell]$. We can easily show that the left and right kernels of C and Z are pairwise in bijection:

Lemma 4.15.

$$\operatorname{rank} C + (n - \tau) = \operatorname{rank} Z$$

Furthermore, let L_C and R_C denote the left respectively right kernel space of C, and L_Z and R_Z that of Z. Then $\dim L_C = \dim L_Z$ and $\dim R_C = \dim R_Z$. Furthermore, the maps $\varphi_L : L_Z \to L_C$ and $\phi_R : R_C \to R_Z$ defined as

$$\begin{aligned} \phi_R\big((q_1, \ldots, q_{\vec{\nabla}_\Sigma^Q})\big) &= (q_{\vec{\nabla}_0^Q+1}, \ldots, q_{\vec{\nabla}_\Sigma^Q}) \\ \varphi_L\big((v_1, \ldots, v_{\tau+1})\big) &= (v(\alpha_1)\zeta_1, \ldots, v(\alpha_n)\zeta_n), \qquad \text{where } v(x) = \textstyle\sum_{i=1}^{\tau+1} v_i x^{i-1} \end{aligned}$$

are bijections.

Proof. Consider first ϕ_R. Considering how we obtained Z, clearly $\dim R_C \geq \dim R_Z$ and the image of ϕ_R is a subset of R_Z. Now ϕ_R is linear so it is a homomorphism, but its kernel is also trivial: otherwise there would exist a non-zero vector $(q_1, \ldots, q_{\vec{\nabla}_0^Q}, 0, \ldots, 0) \in R_C$, by which $C_0(q_1, \ldots, q_{\vec{\nabla}_0^Q})^T = \mathbf{0}$. However, since C_0 is a Vandermonde matrix with more rows than columns, it has full column rank so such a vector can't exist. Thus ϕ_R is injective and we get $\dim R_C \leq \dim R_Z$ from which equality follows, and so ϕ_R is also onto.

Now, since Z has $n - \tau$ fewer columns than C, it follows from the fundamental theorem of linear algebra (applied twice) that $\operatorname{rank} C - \dim R_C = \vec{\nabla}_\Sigma^Q = \operatorname{rank} Z - \dim R_Z + (n - \tau)$, so from the above isomorphism $\operatorname{rank} C = \operatorname{rank} Z + (n - \tau)$. Since Z also has $n - \tau$ fewer rows than C, applying the fundamental theorem to the left kernel spaces immediately then yields $\dim L_C = \dim L_Z$.

Consider now φ_L, and note that the map is equivalent to $\boldsymbol{v} \mapsto \boldsymbol{v} \cdot \big(\operatorname{diag}(\boldsymbol{\zeta})V_{\tau+1}(\boldsymbol{\alpha})\big)^T$, so the image of φ_L must be a subset of L_C. Due to the equality in dimension, φ_L must thus be at the same time onto and injective if it has a trivial kernel; but this must be true, since otherwise there would exists a non-zero polynomial of degree $\tau < n$ with n zeroes. □

At first glance, Z_t looks alien, but we can find a most familiar description of it: from the definition of $S_\bullet^{(t)} = \frac{\overline{R}^{(t)}}{\overline{G}}$ we can expand and rewrite

$$S_\bullet^{(t)}(x) = \frac{\sum_{i=1}^n r_i'^t \zeta_i \prod_{j\neq i} \frac{1-x\alpha_j}{\alpha_i-\alpha_j}}{\prod_{i=1}^n (1-x\alpha_i)} = \sum_{i=1}^n \frac{r_i'^t \zeta_i}{1-x\alpha_i} = \sum_{j=0}^\infty x^j \sum_{i=1}^n r_i'^t \zeta_i \alpha_i^j$$

So Z_t is the first $n-t(k-1)-1$ coefficients of $S_\bullet^{(t)}$ arranged in Hankel form! With the above lemma, this leads quickly to the following

Lemma 4.16. *A vector $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_\tau)$ is in the left kernel of Z if and only if $\overline{\lambda}(x) = \sum_{i=0}^\tau \lambda_{\tau-i} x^i$ is a solution to the Power Syndromes 2D key equation of Corollary 4.8.*

Proof. Assume that $\boldsymbol{\lambda} Z = \mathbf{0}$ so for $t \geq 1$ then $\boldsymbol{\lambda} Z_t = \mathbf{0}$, which means

$$\sum_{j=0}^\tau \lambda_j s_{j+i} = 0, \quad i = 0, \ldots, \grave{\nabla}_t^Q - 1$$

where $s_h = \sum_{i=1}^n r_i'^t \zeta_i \alpha_i^h$ and is the hth coefficient of $S_\bullet^{(t)}(x)$. Since $\overline{\lambda}(x)$ consists of the elements of $\boldsymbol{\lambda}$ in reverse order, then $\sum_{j=0}^\tau \lambda_j s_{j+i}$ is the $(i+\tau)$th coefficient of the product $\overline{\lambda}(x) S_\bullet^{(t)}(x)$; so the above states $\deg(\overline{\lambda}(x) S_\bullet^{(t)}(x) \bmod x^{\grave{\nabla}_t^Q - 1 + \tau}) < \tau$. Since $\grave{\nabla}_t^Q - 1 + \tau = n - t(k-1) - 1$, this immediately implies that $\overline{\lambda}(x)$ is a solution to the 2D key equation of Corollary 4.8. The opposite direction runs analogously when assuming that $\overline{\lambda}(x)$ is solution to the 2D key equation and has degree at most τ. □

From the above two lemmas we immediately get

Theorem 4.17. *For some number of occurred errors, $|\mathcal{E}| \leq \tau$, the probability that Power Syndrome decoding fails is upper bounded by the probability that Coppersmith–Sudan decoding fails.*

Proof. If Power Syndrome decoding fails, then there is a minimal solution $\overline{\lambda}(x) \neq \overline{\Lambda}(x)$ to the 2D key equation of Corollary 4.8 with $\deg \overline{\lambda} \leq |\mathcal{E}| \leq \tau$. By Lemma 4.16, there corresponds to this a vector in the left kernel of Z, and according to Lemma 4.15, this gives a vector in the left kernel of C; studying the two bijections, note that this vector is $\big(\overline{\lambda}(\alpha_1)\zeta_1, \overline{\lambda}(\alpha_2)\zeta_2, \ldots, \overline{\lambda}(\alpha_n)\zeta_n\big)$. Since $\Lambda \nmid \lambda$ then obviously this vector must be non-zero in one or more of the error positions; hence Coppersmith–Sudan decoding fails. □

Remark. Note that the probabilities of failing are not the same: if Coppersmith–Sudan decoding fails we could indeed go back through the bijections to a polynomial $\overline{\lambda}(x)$ which is a solution to the Power Syndrome key equation, but this solution need not be minimal! In this case Power Syndrome decoding wouldn't fail. Simulations that I have performed with Codinglib [Nie13a] on small codes indicate that the difference here is small but measurable, so these cases really do crop up randomly. ◆

Remark. Theorem 4.17 together with Proposition 4.14 implies that the probability that Power decoding (Gao or Syndrome) fails when at most τ errors has occured is upper bounded by $1 - (1 - \frac{\ell}{q})^{|\mathcal{E}|}$. This is not a very impressive bound, though, and it's a long-shot away from the much more quickly decaying behaviours discussed on page 103.

Simulations I have performed indicate much lower failure probabilities; in particular, for $n = q - 1 = 255$ and for $|\mathcal{E}|$ just a few errors from the upper bound τ, the probability is close enough to 0 for such instances never cropping up in many thousands of experiments for both Coppersmith–Sudan as well as Power decoding.

My hope is that it is possible to improve the upper bound on the failure probability of Coppersmith–Sudan, owing to its rather direct linear algebraic nature, and that this could then give a satisfactory upper bound on the failure probability of Power decoding. As an entry to this, Coppersmith and Sudan's failure probability depends on the number of roots in a series of polynomials; these polynomials are neither independent nor completely random, but fairly so. The derived failure probability is based on the worst case: that all of these polynomials completely split over $\mathbb{F}_q$ with distinct roots. However, it has been shown that a polynomial in $\mathbb{F}_q[x]$ on average has only a single root [Leo06], and if one is able to use this estimate unrestrictedly, we would get the failure probability $1 - (1 - \frac{1}{q})^{|\mathcal{E}|}$. This is, however, still not nearly as good as simulations predict, and does not decay fast enough for $|\mathcal{E}|$ moving away from the decoding upper bound. ◆

Remark. We remarked earlier that in the right kernel of C are the vectors forming Q polynomials of Sudan decoding. Similarly, it is straightforward to show that in the right kernel of Z are the $\overline{Q}_1, \ldots, \overline{Q}_\ell$ sought in the Roth–Ruckenstein speedup Section 3.4.1. In a precise sense, then, the left kernel of Z contains vectors forming error locators and the right kernel contains vectors forming Q polynomials; this is yet another facet of the "orthogonality" between Power Syndrome and Roth–Ruckenstein decoding, as discussed in the previous section.

In fact, one can use a similar matrix-style rewrite from C to Z to obtain the Q-finding key equations from Section 3.4 also for $s > 1$; Ruckenstein did so in her thesis [Ruc01, Section 5.2 and 5.3], and Beelen and Høholdt did the analogous rewrite for AG codes [BH08b]. ◆

4.4 Power decoding of Hermitian codes

Several classes of codes have minimum distance decoders based on solving a key equation containing the error locator: these include Goppa codes, AG codes, L, G codes and Gabidulin codes. One can in each instance consider whether the principle of power decoding can be applied as well; this is unknown for three of the above families, but for at least some AG codes, it does apply in a straightforward manner; this was demonstrated by Kampf [Kam11]. The main problem lies in how to solve

the emerging "set of key equations", since they must inherently be defined over the function field of the AG code.

Kampf handled only the one-point Hermitian codes. In [Kam11], only a trivial, slow algorithm was given for solving the key equations. In [KL13], Kampf and Li instead used an extension of the Euclidean algorithm to bivariate polynomials, which is probably more efficient, but see discussion in Section 4.5. We will show how to map the set of key equations of the function field into a larger set over $\mathbb{F}[x]$, and this will yield a 2D key equation as defined in Section 2.5. Using Mulders–Storjohann to solve this equation is quite related to Kampf and Li's Euclidean algorithm extension, while the other choices of module minimisation yield better performance.

Though the approach most likely extends to many other AG codes, we will like in Section 3.5—and like Kampf—only handle the one-point Hermitian codes.

Like in the beginning of this chapter, we will base our exposition on a "Gao key equation", where earlier work for decoding using a key equation has been a syndrome-style formulation, e.g. [JLJ+89, JLJH92, SJH95, OBA08, Kam11, BK12]. To the best of my knowledge, a Gao-style key equation has not been proposed for decoding AG codes previously. Apart from the joy of variety, the Gao-style formulation admits succinct arguments based on Riemann–Roch-spaces, which makes the derivation of the central result, Proposition 4.20, quite short. Furthermore, as for the case of GRS codes, solving the key equation of the Gao formulation reveals the information polynomial f directly, with no need for finding error values and correcting. It should perhaps be noted that the basic Gao key equation, i.e. $\ell = 1$, can be seen as just a facet of the Sudan decoder for AG codes [SW98] with $\ell = 1$, in a manner analogous to what we described in the beginning of Section 4.3.1.

We will use all the notation introduced in Section 3.5.1; consult possibly Appendix A. In particular, we will be decoding the codes of Definition 3.48 on page 80 so consider such a code $\mathcal{C}$. Let as usual $\boldsymbol{c} \in \mathcal{C}$ be some sent codeword and $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ be the received word which has been subjected to some noise $\boldsymbol{e} \in \mathbb{F}_{q^2}^n$. Let also $\mathcal{E} = \{i \mid e_i \neq 0\}$.

Recall G and R from Definition 3.54 on page 83, and extend the latter to its powers:

$$R^{(t)} \neq 0 : \; R^{(t)}(P_i) = r_i^t \quad \forall i = 1, \ldots, n, \quad t \in \mathbb{N}_0 \tag{4.7}$$

Again, these can be found by solving the emerging linear systems of equations or using the explicit formula of Lemma 3.50 on page 81. Now for the error locator. Again, we are forced to be less explicit than in the case of GRS codes:

Definition 4.18. The *error locator* Λ is the non-zero polynomial in $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$ with minimal $\deg_{\mathcal{H}}$ and $\operatorname{LC}_{\mathcal{H}}(\Lambda) = 1$.

Clearly, $\Lambda \in \mathit{Я}$ since the defining Riemann–Roch space is a subset of $\mathit{Я}$. The definition is well-defined, i.e. there is exactly one element in the Riemann–Roch space satisfying the restrictions: for if we have two error locators Λ_1, Λ_2 with the same minimal $\deg_{\mathcal{H}}$, then $\operatorname{LM}_{\mathcal{H}}(\Lambda_1) = \operatorname{LM}_{\mathcal{H}}(\Lambda_2)$ since all $x^i y^j$ have unique $\deg_{\mathcal{H}}$

whenever $j < q$. But then $\deg_{\mathcal{H}}(\Lambda_1 - \Lambda_2) < \deg_{\mathcal{H}} \Lambda_1$, and this linear combination must be in $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$. Since Λ_1 has minimal order of non-zero elements in this Riemann-Roch space, then $\Lambda_1 - \Lambda_2 = 0$.

Lemma 4.19. $|\mathcal{E}| \leq \deg_{\mathcal{H}} \Lambda \leq |\mathcal{E}| + g$

Proof. Being in $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$ specifies $|\mathcal{E}|$ homogeneous equations in the coefficients of Λ, so by Lemma 3.47 on page 80, we will still have more coefficients than equations after requiring $\deg_{\mathcal{H}} \Lambda < |\mathcal{E}| + g + 1$. For the lower bound, then since $\deg(-\sum_{i\in\mathcal{E}} P_i + tP_\infty) < 0$ for $t < |\mathcal{E}|$ we must have $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + tP_\infty) = \{0\}$ whenever $t < |\mathcal{E}|$. Since $\Lambda \neq 0$ is in this Riemann–Roch space when $t = \deg_{\mathcal{H}} \Lambda$, then clearly $\deg_{\mathcal{H}} \Lambda \geq |\mathcal{E}|$. □

We then surprisingly easily arrive at the analogue of Corollary 4.3 on page 96:

Proposition 4.20. $\Lambda R^{(t)} \equiv \Lambda f^t \mod G$ *for* $t \in \mathbb{N}_0$ *as a congruence over* $\mathcal{R}$.

Proof. We must have $\Lambda R^{(t)} - \Lambda f^t = \Lambda(R^{(t)} - f^t) \in \mathcal{L}(-\sum_{i=1}^n P_i + \infty P_\infty)$ since for $i \in \mathcal{E}$ then $\Lambda(P_i) = 0$ while for $i \notin \mathcal{E}$ then $R^{(t)}(P_i) = f^t(P_i)$. Recall that $(G) = \sum_{i=1}^n P_i - nP_\infty$; therefore by Lemma 3.46 on page 80 we must have $G \mid \Lambda(R^{(t)} - f^t)$ over $\mathcal{R}$. □

This means that the sought Λ is a, hopefully, quite small solution to a list of key equations – but over $\mathcal{R}$. Just as in the case of Power decoding GRS codes, we will solve these key equations by regarding the right-hand sides as unknowns independent of Λ; therefore, only the first few values of t will provide us with new information. In particular, when $\deg_{\mathcal{H}}(\Lambda f^t) > \deg_{\mathcal{H}}(G)$ then the key equation becomes worthless to us. We do not know know $\deg_{\mathcal{H}} \Lambda$ but we can at least stop once $tm \geq \deg_{\mathcal{H}} f^t > \deg_{\mathcal{H}} G = q(n/q) = n$. Thus, in the following, assume that $\ell \in \mathbb{Z}_+$ is a parameter chosen such that $\ell m < n$.

For us to be able to apply the methods of Chapter 2, and in particular model the equations as in Problem 2.31 on page 26, we need somehow to map it into key equations over $\mathbb{F}_{q^2}[x]$.

Theorem 4.21. *Let* $\Lambda(x, y) = \sum_{i=0}^{q-1} y^i \Lambda_i(x)$. $(\Lambda_0, \ldots, \Lambda_{q-1})$ *is a solution to the 2D key equation of Type 2 with parameters:*

- $\rho = q$ *and* $\sigma = q\ell$.
- $G_j^{\mathrm{KE}} = G$ *for* $j = 1, \ldots, q\ell$.
- $S_{i,j} = (m_{i,h}^{R^{(t)}} \bmod G)$ *where* $t = \lfloor (j-1)/q \rfloor + 1$ *and* $h = ((j-1) \bmod q) + 1$, *and* $m_{i,h}^{R^{(t)}}$ *is as in (4.8) and (4.9) on page 115.*
- $\nu = q$.
- $\eta_i = (i-1)(q+1) + \ell m + 1$ *for* $i = 1, \ldots, q$.
- $w_j = w_{t,h}$ *for* $j = 1, \ldots, q\ell$, *where* $w_{t,h} = (h-1)(q+1) + (\ell - t)m$ *and* t, h *are as above.*

Proof. The key equation of Proposition 4.20 implies for each t that there is a $p_t(x, y) \in \mathcal{R}$ such that $\Lambda(x, y)R^{(t)}(x, y) - \Lambda(x, y)f^t(x, y) = p_t(x, y)G(x)$. In par-

ticular, the above holds if $\Lambda(x,y)R^{(t)}(x,y)$, $\Lambda(x,y)f^t(x,y)$ and $p_t(x,y)G(x)$ are all represented fully reduced by $\mathcal{H}$. Since $G(x)$ is exactly only a polynomial in x, then $p_t(x,y)G(x)$ is reduced when $p_t(x,y)$ is. Therefore, for each of the q powers of y, the key equation can be regarded *independently* as key equations over $\mathbb{F}_{q^2}[x]$, once both sides have been fully reduced by $\mathcal{H}$. Recalling the notation on page 84 for dealing with elements of $\mathcal{R}$ as vectors, then denoting $\boldsymbol{\Lambda} = \Upsilon(\Lambda)$ we have

$$\boldsymbol{\Lambda}\Pi(R^{(t)})\Xi \equiv \boldsymbol{\Lambda}(\Pi(f)\Xi)^t \mod G(x)$$

where the modulo operation is over $\mathbb{F}_{q^2}[x]$ and independent on each position of the vectors. If we let $\boldsymbol{m}_{:,h}^{R^{(t)}}$ be the hth column of $\Pi(R^{(t)})\Xi$ and $\boldsymbol{m}_{:,h}^{f^t}$ the hth column of $(\Pi(f)\Xi)^t$ for $h = 1, \ldots, q$, the above vectorised key equation expands to:

$$\boldsymbol{\Lambda}\boldsymbol{m}_{:,h}^{R^{(t)}} \equiv \boldsymbol{\Lambda}\boldsymbol{m}_{:,h}^{f^t} \mod G(x), \quad h = 1, \ldots, q$$

Explicitly, letting $\mathbf{0}_t$ denote a sequence of t zeroes, then the entries of $\boldsymbol{m}_{:,h}^{R^{(t)}}$ satisfy for $1 \le i \le q-1$ and $2 \le h \le q$:

$$\begin{aligned} m_{i,h}^{R^{(t)}} &= \left(\mathbf{0}_{i-1},\ R_0^{(t)},\ \ldots,\ R_{q-1}^{(t)},\ \mathbf{0}_{q-i+1}\right) \cdot \left(\mathbf{0}_{h-1},\ 1,\ \mathbf{0}_{q-2},\ -1,\ x^{q+1},\ \mathbf{0}_{q-h-1}\right) \\ &= \begin{cases} R_{h-i}^{(t)} & \text{if } i < h \\ R_0^{(t)} - R_{q-1}^{(t)} & \text{if } i = h \\ -R_{q-1-(i-h)}^{(t)} + x^{q+1} R_{q-(i-h)}^{(t)} & \text{if } i > h \end{cases} \end{aligned} \tag{4.8}$$

For $h = 1$ we instead have

$$\boldsymbol{m}_{:,1}^{R^{(t)}} = \left(R_0^{(t)},\ x^{q+1}R_{q-1}^{(t)},\ x^{q+1}R_{q-2}^{(t)},\ \ldots,\ x^{q+1}R_1^{(t)}\right) \tag{4.9}$$

Thus, for each t, then $\boldsymbol{\Lambda}$ must be a solution to the 2D key equation with q equations and $S_{i,j} = m_{i,h}^{R^{(t)}}$, and with a right-hand side of $\Psi_{t,h}(x) = \boldsymbol{\Lambda}\boldsymbol{m}_{:,h}^{f^t}$ for $h = 1, \ldots, q$, for some assignment of the weights. Clearly, it will make no difference reducing $m_{i,j}^{R^{(t)}}$ modulo G. Note that in the theorem we have identified the single index $j = 1, \ldots, q\ell$ with the double index (t,i) by $j = q(t-1)+i$.

We have then left only to show that the choices of weights are correct, i.e. that for each $t = 1, \ldots, \ell$:

$$\begin{aligned} \max_{i=1,\ldots,q} \{q \deg \Lambda_i + (i-1)(q+1) + \ell m + 1\} > \\ \max_{h=1,\ldots,q} \{q \deg \Psi_{t,h} + (h-1)(q+1) + (\ell - t)m\} \end{aligned} \tag{4.10}$$

However, by the way we derived it, $\Psi^{(t)}(x,y) = \sum_{h=0}^{q-1} y^h \Psi_{t,h}(x) = \Lambda(x,y)f^t(x,y)$ after reduction by $\mathcal{H}$. Therefore $\deg_{\mathcal{H}} \Psi^{(t)} = \deg_{\mathcal{H}}(\Lambda f^t) \le \deg_{\mathcal{H}} \Lambda + tm$. But clearly $\deg_{\mathcal{H}} \Psi^{(t)} = \max_h\{q \deg \Psi_{t,h}(x) + (h-1)(q+1)\}$. Therefore the right-hand side of (4.10) is at most $\deg_{\mathcal{H}} \Lambda + \ell m$. Similarly, the left-hand side is *exactly* $\deg_{\mathcal{H}} \Lambda + \ell m + 1$. This finishes the proof. □

The matrix to be minimised for solving a 2D key equation in general was given in (2.3) on page 30; for the 2D key equation of Theorem 4.21, the matrix will therefore have the form:

$$M_{\mathcal{H}} = \left(\begin{array}{c|c|c|c|c} I_{q\times q} & [M_{R^{(1)}}\Xi]_G & [M_{R^{(2)}}\Xi]_G & \dots & [M_{R^{(\ell)}}\Xi]_G \\ \hline \mathbf{0} & \multicolumn{4}{c}{G(x)I_{(q\ell)\times(q\ell)}} \end{array}\right) \in \mathbb{F}_{q^2}[x]^{(q(\ell+1))\times(q(\ell+1))} \tag{4.11}$$

where $[M_{R^{(t)}}\Xi]_G$ equals $M_{R^{(t)}}\Xi$ where each entry has been reduced modulo G.

Just as for Power decoding of GRS codes, the above leads immediately to a decoding algorithm, which will most of the time—but not always—be capable of decoding beyond what is possible with only a single key equation, i.e. $\ell = 1$. This case is more difficult to analyse than for Power decoding of GRS codes, so we have split the analogue of Proposition 4.4 on page 97 into several results. First a decoding radius lower bound; complications pertaining to $\deg_{\mathcal{H}} \Lambda$ usually exceeding $|\mathcal{E}|$ due to the positive genus of the curve leads to a *defect* in decoding using a single key equation: one can decode somewhat less than half the minimum distance!

Proposition 4.22. *In the context of Theorem 4.21, the vector* $(\Lambda_0, \dots, \Lambda_{q-1})$ *is a minimal solution to the 2D key equation whenever* $|\mathcal{E}| \leq \frac{d-1}{2} - \frac{g}{2}$.

Proof. Let $(\lambda_0, \dots, \lambda_{q-1}, \psi_0, \dots, \psi_{q-1})$ be a minimal solution to the 2D key equation for $\ell = 1$ while $|\mathcal{E}| \leq \frac{n-k}{2} - \frac{g}{2}$, and we will show that $\lambda_i = \gamma\Lambda_i$ for some $\gamma \in \mathbb{F}_{q^2}^*$. Since for $\ell > 1$ we impose further restrictions on the solution set, the analogous statement must then be true. Define $\lambda(x, y) = \sum_{i=0}^{q-1} y^i \lambda_i(x)$ and $\psi(x, y) = \sum_{i=0}^{q-1} y^i \psi_i(x)$. By assumption $\deg_{\mathcal{H}} \lambda \leq \deg_{\mathcal{H}} \Lambda$, and by the degree constraints of the 2D key equation, recalling how we in the proof of Theorem 4.21 established the values of the η_i and w_j, we know

$$\deg_{\mathcal{H}} \lambda + m + 1 > \deg_{\mathcal{H}} \psi \tag{4.12}$$

The 2D key equation is an $\mathbb{F}_{q^2}[x]$ equivalent of the congruence of Proposition 4.20 on page 114 so we can also go back, which means $\lambda R^{(1)} \equiv \psi \mod G$ as a congruence over $\mathcal{R}$, that is, $G \mid (\lambda R^{(1)} - \psi)$. By Lemma 3.46 on page 80 then $\lambda R^{(1)} - \psi \in \mathcal{L}(-\sum_{i=1}^{n} P_i + \infty P_\infty)$. Introduce $\hat{e} = R^{(1)} - f \in \mathcal{R}$ so $\hat{e}(P_i) = e_i$ for $i = 1, \dots, n$. Clearly $\hat{e} \in \mathcal{L}(-\sum_{i\notin\mathcal{E}} P_i + \infty P_\infty)$, which means

$$\lambda f - \psi = (\lambda R^{(1)} - \psi) - \lambda\hat{e} \in \mathcal{L}(-\textstyle\sum_{i\notin\mathcal{E}} P_i + hP_\infty)$$

where h is an upper bound on $\deg_{\mathcal{H}}(\lambda f - \psi)$; by (4.12), we can choose $h = \deg_{\mathcal{H}} \lambda + m$. Now we simply want to show that if $\deg_{\mathcal{H}} \lambda \leq \deg_{\mathcal{H}} \Lambda$ then this Riemann–Roch space is $\{0\}$; for in that case $\lambda f = \psi$, so by the congruence then $G \mid \lambda(R - f)$, which means $\lambda \in \mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$; but Λ has minimal $\deg_{\mathcal{H}}$ of non-zero elements in this Riemann–Roch space, and so we have the sought.

We have $\mathcal{L}(-\sum_{i\notin\mathcal{E}} P_i + hP_\infty) = \{0\}$ at least when the defining divisor is negative, and since all P_i and P_∞ are rational, this happens when $n - |\mathcal{E}| > h = \deg_{\mathcal{H}} \lambda + m$. Now $\deg_{\mathcal{H}} \lambda \leq \deg_{\mathcal{H}} \Lambda \leq |\mathcal{E}| + g$ by Lemma 4.19. Therefore, the divisor is negative at least when

$$n - |\mathcal{E}| > |\mathcal{E}| + g + m \qquad \iff \qquad |\mathcal{E}| < \frac{n-m-g}{2} = \frac{d-g}{2} \qquad \square$$

The upper bound, i.e. when we can be quite certain Power decoding will fail, is somewhat more technical to derive; it is a matter of properly counting linear equations and indeterminates, and we will draw upon a result from Section 2.5.1 and follow the same strategy as the GRS analogue (4.3) on page 97.

Lemma 4.23. *In the context of Theorem 4.21, the vector* $(\Lambda_0, \ldots, \Lambda_{q-1})$ *is* not *a minimal solution whenever* $\deg_{\mathcal{H}} \Lambda + \ell m + 1 > \delta$*, where* δ *is the smallest integer satisfying*

$$\sum_{t=1}^{\ell} \sum_{h=1}^{q} \operatorname{pos}(q^{-1}n - \lceil q^{-1}(\delta - w_{t,h})\rceil) \leq \sum_{i=1}^{q} \lfloor q^{-1}(\delta - \eta_i)\rfloor + q - 1, \tag{4.13}$$

except possibly under special circumstances pertaining to a certain system of equations specified in the proof.

Proof. Let $\boldsymbol{s} = (s_1, \ldots, s_{q(\ell+1)})$ be a minimal solution to the 2D key equation, and let $\bar{\boldsymbol{w}} = (\eta_1, \ldots, \eta_q, w_1, \ldots, w_{q\ell})$, where the η_i and w_j are as in Theorem 4.21. We will prove that $\deg \Phi_{q,\bar{\boldsymbol{w}}}(\boldsymbol{s}) \leq \delta$, where δ is as in the lemma, and this will lead to the sought: since $\boldsymbol{s} = (\Lambda_0, \ldots, \Lambda_{q-1}, \Psi_1, \ldots, \Psi_{q\ell})$ for some Ψ_j is a solution and the 2D key equation is of Type 2, then

$$\deg \Phi_{q,\bar{\boldsymbol{w}}}(\boldsymbol{s}) = \max_{i=1,\ldots,q} \{\nu \deg \Lambda_{i-1} + \eta_i\} = \deg_{\mathcal{H}} \Lambda + \ell m + 1$$

Clearly, if $\delta < \deg_{\mathcal{H}} \Lambda + \ell m + 1$, then $(\Lambda_0, \ldots, \Lambda_{q-1})$ can't be a minimal solution. But the claim on $\boldsymbol{s}$ follows immediately from Proposition 2.39 on page 33: this states that, except in special circumstances, then $\deg \Phi_{q,\bar{\boldsymbol{w}}}(\boldsymbol{s}) \leq \delta$ where δ is the least integer satisfying

$$\sum_{j=1}^{\sigma} \operatorname{pos}(\deg G_j^{\mathrm{KE}} - \lceil \nu^{-1}(\delta - w_j)\rceil) \leq \sum_{i=1}^{\rho} \lfloor \nu^{-1}(\delta - \eta_i)\rfloor + (\rho - 1)$$

Inserting $\nu = \rho = q$, $\sigma = q\ell$ and $\deg G_j^{\mathrm{KE}} = n/q$, and changing the left sum to a double sum in t and h, we get the sought. Note that the exception to the bound of the lemma is exactly the exception also stated in Proposition 2.39. $\square$

For any given code parameters, the δ of Lemma 4.23 is easy to compute; however, analytically it is a bit hard to work with. It turns out that we can lower bound δ rather precisely:

Proposition 4.24. *In the context of Lemma 4.23, then*

$$\delta \geq \tfrac{\hat{\ell}}{\hat{\ell}+1}n + (\ell - \tfrac{1}{2}\hat{\ell})m + g + \tfrac{1}{\hat{\ell}+1}$$

where $\hat{\ell} = \min\left\{\ell\ ;\ \left\lfloor -(\tfrac{1}{2}+\tfrac{g}{m}) + \sqrt{(\tfrac{1}{2}+\tfrac{g}{m})^2 + \tfrac{2(n-g-1)}{m}}\right\rfloor\right\}$.

Proof. Overall, we will progress in a manner similar to in the proof of Corollary 2.40 on page 34 combined with the proof of Proposition 4.4 on page 97, but we first need to handle the positive operator and the rounding operators.

Consider the least integer $\hat{\delta}$ which satisfies

$$\sum_{t=1}^{\ell} \flat_t \sum_{h=1}^{q} (q^{-1}n - \lceil q^{-1}(\hat{\delta} - w_{t,h})\rceil) \leq \sum_{i=1}^{q} \lfloor q^{-1}(\hat{\delta} - \eta_i)\rfloor + q - 1 \tag{4.14}$$

where $\flat_t = 1$ if for *all* $h = 1, \ldots, q$ it holds that $q^{-1}n \geq \lceil q^{-1}(\hat{\delta} - w_{\hat{\ell},h})\rceil$, and $\flat_t = 0$ otherwise. Clearly then $\hat{\delta} \leq \delta$. Our estimation will be of $\hat{\delta}$.

Note first that there is an index $\hat{\ell}$ such that $\flat_t = 1$ for all $t \leq \hat{\ell}$ and $\flat_t = 0$ for $t > \hat{\ell}$ since $w_{t,h} > w_{t+1,h}$ for all t and h. Now we can get rid of the rounding operators: since $\hat{\delta} - \eta_i = (\hat{\delta} - \ell m - 1) - (i-1)(q+1)$, then for $i = 1, \ldots, q$, this expression goes through all congruence classes modulo q. Therefore, when rounding these down after division by q, we are essentially subtracting $0, \frac{1}{q}, \frac{2}{q}, \ldots, \frac{q-1}{q}$ in some order; thus, $\sum_{i=1}^{q} \lfloor q^{-1}(\hat{\delta} - \eta_i)\rfloor = \sum_{i=1}^{q} q^{-1}(\hat{\delta} - \eta_i) - q^{-1}\binom{q}{2}$. Similarly, for each t then $d - w_{t,h}$ runs through all congruence classes modulo q for $h = 1, \ldots, q$, so the ceiling-function in the inner loop subtracts exactly $q^{-1}\binom{q}{2}$ in each of the outer loop iterations. Inserting this in (4.14), we get:

$$q^{-1}\sum_{t=1}^{\hat{\ell}} \left(-\tbinom{q}{2} + \sum_{h=1}^{q} (n - (\hat{\delta} - w_{t,h}))\right) \leq \sum_{i=1}^{q} q^{-1}(\hat{\delta} - \eta_i) - q^{-1}\tbinom{q}{2} + q - 1 \iff$$
$$\hat{\delta} \geq \tfrac{\hat{\ell}}{\hat{\ell}+1}n + (\ell - \tfrac{1}{2}\hat{\ell})m + g + \tfrac{1}{\hat{\ell}+1} \tag{4.15}$$

So $\hat{\delta}$ is the smallest integer at least the above right-hand side. Unfortunately, $\hat{\delta}$ depends on $\hat{\ell}$ and vice versa; we handle this in the same manner as in the proof of Corollary 2.40.

Note first that since $q^{-1}n$ is an integer, then $q^{-1}n \geq \lceil q^{-1}(\hat{\delta} - w_{t,h})\rceil$ is equivalent to $q^{-1}n \geq q^{-1}(\hat{\delta} - w_{t,h})$, i.e. $n + w_{t,h} \geq \hat{\delta}$. Therefore, since $w_{t,h} < w_{t,h+1}$, then $\hat{\ell}$ can be characterised simply as the greatest index such that $n + w_{\hat{\ell},0} \geq \hat{\delta}$. Define now $\zeta_t = n + w_{t,0}$ for $t = 1, \ldots, \ell$ as well as $\zeta_{\ell+1} = -\infty$, and note that $\zeta_1 > \zeta_2 > \ldots > \zeta_{\ell+1}$. Define also $\delta_t = \frac{t}{t+1}n + (\ell - \frac{1}{2}t)m + g + \frac{1}{t+1}$ for $t = 1, \ldots, q$. From these definitions, it is clear that

$$\zeta_{\hat{\ell}+1} < \hat{\delta} = \lceil \delta_{\hat{\ell}}\rceil \leq \zeta_{\hat{\ell}}$$

Since the ζ_t are integers, then $\zeta_{\hat{\ell}+1} < \lceil \delta_{\hat{\ell}} \rceil$ is equivalent to $\zeta_{\hat{\ell}+1} < \delta_{\hat{\ell}}$. Similarly, $\lceil \delta_{\hat{\ell}} \rceil \leq \zeta_{\hat{\ell}}$ is equivalent to $\delta_{\hat{\ell}} \leq \zeta_{\hat{\ell}}$. Assume first that $\hat{\ell} < \ell$. Inserting the expression of $\delta_{\hat{\ell}}$ into these inequalities and rearranging, we arrive at

$$-(\chi+1) + \sqrt{\chi^2 + 2\chi + \tfrac{2(n-g-1)}{m}} < \hat{\ell} \leq -\chi + \sqrt{\chi^2 + \tfrac{2(n-g-1)}{m}}$$

where $\chi = \frac{1}{2} + \frac{g}{m}$. If the lower bound is greater than ℓ, then the assumption $\hat{\ell} < \ell$ must have been wrong, but then $\hat{\ell} = \ell$: this choice is also valid, since obviously the upper bound must be satisfied implying $\delta_\ell < \zeta_\ell$, and therefore $-\infty = \zeta_{\ell+1} < \delta_\ell \leq \zeta_\ell$.

On the other hand, if the lower bound is smaller than ℓ, then $\hat{\ell}$ must be in the above interval, which must contain an integer since $\hat{\ell}$ was well-defined. The interval has width less than 1, so this integer must be the floor of the upper bound, exactly as stated in the proposition. □

The purpose of the last two results is to estimate how many errors we will *usually* be able to correct. Though we have no bound on the failure probability within the two extreme ends, the prediction from linear algebra on when the 2D key equation ought to have a solution provides a good intuition, and simulations indicate that this describes the truth to a large extend: linear systems of equations only have unexpected small solutions when the system has surprisingly low rank; and for this the errors should probably be made in a very "unlucky" way. Whenever $\deg_{\mathcal{H}} \Lambda$ is below the upper bound of Lemma 4.23, we would therefore expect no shorter solutions to the key equation. The lower bound on δ from Proposition 4.24 therefore provides an estimate on the "usual" decoding radius we should expect from the Power decoding of Hermitian codes: for most error patterns then $\deg_{\mathcal{H}} \Lambda = |\mathcal{E}| + g$ so decoding should usually succeed only whenever:

$$|\mathcal{E}| \leq \tfrac{\hat{\ell}}{\hat{\ell}+1} n - \tfrac{1}{2}\hat{\ell} m - \tfrac{\hat{\ell}}{\hat{\ell}+1} \tag{4.16}$$

Note that most of these concerns are completely analogous to Power decoding of GRS codes. Note also how the above decoding bound matches *exactly* that of Power decoding, Proposition 4.4 on page 97, since m corresponds to $k-1$. Also analogous to this case, it seems possible that if more errors than the above occur, but they align in a particular *lucky* way, then we might still be able to decode them; this seems to happen quite rarely in practice, though, see Example 4.25 on page 120.

Remark. For $\ell = 1$, i.e. minimum distance decoding, then (4.16) gives $|\mathcal{E}| \leq \frac{n-m-1}{2} = \frac{d-1}{2}$, while Proposition 4.22 only promises $|\mathcal{E}| \leq \frac{d-1-g}{2}$. This is an interesting caveat of "pure" key equation decoding AG codes: we are only *assured* decoding success until $g/2$ less than $(d-1)/2$, but *almost always*, decoding will succeed all the way until $(d-1)/2$. Unfortunately and surprisingly, I do not know of any work investigating the probability that we will fail.

Still having $\ell = 1$, it is even possible that we should be able to decode *beyond* $\frac{d-1}{2}$: the error positions might align such that $\deg_{\mathcal{H}} \Lambda < |\mathcal{E}| + g$, or the rare linear algebraic circumstances indicated by Lemma 4.23 might occur. The probability of

the former event occurring *has* been investigated: for low rate codes it was proven to be $1/q$ asymptotically in [JNH99], and in [Han01] the same was proven for all rates. ◆

Remark. Clearly, $\hat{\delta} \leq \delta$, where $\hat{\delta}$ is from the proof of Proposition 4.24, but they should also be quite close to each other; in fact, if $m \geq q^2 - 1 = 2g + q$, then $w_{t,h} \geq w_{t+1,h'}$ for all $t < \ell$ and h, h'; so the contributions for $t = \hat{\ell} + 2, \ldots, \ell$ in (4.13) on page 117 will indeed be zero, while for $t = \hat{\ell}+1$ it might be slightly positive, but still small.

Just as in the case of Power Gao decoding, in Proposition 4.24 it usually does not make sense to choose ℓ high enough that $\ell > \hat{\ell}$; however, since $\hat{\delta}$ is just an estimate on δ, it just might and one should for concrete parameters calculate the true value of δ for various ℓ. ◆

Example 4.25. *Consider $q = 7$ and the Hermitian code with parameters $[q^3 = 343,\ 35,\ \geq 288]$ and $m = 55$. This code was chosen to be somewhat comparable with the [250, 70, 181] GRS code of Example 3.5 on page 50. They have similar codeword lengths in bits and similar relative minimum distance, which of course means that the Hermitian code has lower rate.*

For this curve $g = 21$, so by Proposition 4.22, we are assured of decoding success whenever $|\mathcal{E}| \leq \lfloor \frac{d-1-g}{2} \rfloor = 133$, but with $\ell = 1$ we would usually be able to decode up to $\lfloor \frac{d-1}{2} \rfloor = 143$ errors. Proposition 4.24 sets $\ell = \min\{\ell; \lfloor 2.65 \rfloor\}$, so by choosing $\ell = 2$ the proposition promises $\delta \geq 305$ and one sees that equality indeed holds by solving (4.13) on page 117. By (4.16), this gives a "usual" decoding radius of 173 errors. However, this is not quite optimal: with $\ell = 3$ Proposition 4.24 would set $\hat{\ell} = 2$ and promise $\delta \geq 360$; but solving (4.13), we get the true $\delta = 363$: this yields a slightly better decoding radius of 174.

Nevertheless, let us choose $\ell = 2$ and decode 173 errors. $M_{\mathcal{H}}$ from (4.11) on page 116 is then a 21×21 matrix over $\mathbb{F}_{7^2}[x]$, and the weights of the 2D key equation are set as:

$$\boldsymbol{\eta} = [111, 119, 127, 135, 143, 151, 159]$$
$$\boldsymbol{w} = [53, 63, 71, 79, 87, 95, 103, 0, 8, 16, 24, 32, 40, 48]$$

To find a minimal solution, we will minimise $\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}})$; in a usual run, this matrix has the following ν^{-1}-normalised characteristics:

$$q^{-1}\text{maxdeg}\left(\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}})\right) = 63\tfrac{5}{7} \qquad q^{-1}\Delta(\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}})) = 304$$

Compare these figures with those for Example 4.5 on page 99 of Power Gao decoding GRS codes. The orthogonality defects are very comparable, so we would expect the minimisation algorithms Mulders–Storjohann, Alekhnovich and the Demand–Driven to run a similar number of iterations. Though the matrix for decoding Hermitian codes is larger by a factor $q = 7$ in each dimension, so is the normalised max-degree of the matrix more than a factor 5 smaller. Of course, since the codes are

not completely comparable, and since ℓ is only 2 in this example but 3 in the Gao example, direct comparisons can be misleading.

The decoding method has been implemented and tested in Codinglib [Nie13a]. Solving the above 2D key equation for various weights of random error patterns with 500 trials each yielded the following rates of decoding success:

# Errors	*172*	*173*	*174*
Success rate	*100%*	*98%*	*2%*

Remember that two types of events can make the method decode beyond the expected 173 errors: Λ has $\deg_{\mathcal{H}}$ lower than the expected $|\mathcal{E}| + g = 195$, or the linear system of equations one can set up to solve the 2D key equation is degenerate in the way specified in Proposition 2.39. In all of the 10 successful decodings in the above test, the first of these was the case. ♠

Proposition 4.26. *The worst-case complexity of finding Λ and f as a minimal solution to the 2D key equation of Theorem 4.21, or fail, is as in Table 4.3, for various choices of module minimisation algorithm.*

Proof. Let $\bar{\boldsymbol{w}} = (\eta_1, \ldots, \eta_q, w_1, \ldots, w_{q\ell})$, where the η_i and w_j are as in Theorem 4.21. The proposition follows from Table 2.4 on page 42 after estimating the measures of $\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}})$. We apply Lemma 2.38 on page 32: we easily see $\max\{\eta_i\} = (q-1)(q+1) + \ell m + 1$, while $\max\{\nu G_j + w_j\} = q(n/q) + (q-1)(q+1) + (\ell-1)m$, and so the ν-normalised max-degree becomes:

$$\gamma = \nu^{-1}\mathrm{maxdeg}\left(\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}})\right) = q^{-1}(n + q^2 + (\ell-1)m - 1) \in O(n/q + q)$$

From this and again Lemma 2.38, we immediately get $\delta = \nu^{-1}\Delta(\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}}))$ upper bounded by

$$\begin{aligned} &q^{-1}\big(q\big(\mathrm{maxdeg}\left(\Phi_{q,\bar{\boldsymbol{w}}}(M_{\mathcal{H}})\right) - 1\big) - \textstyle\sum_{i=1}^{\rho}\eta_i\big) \\ &\quad = n + q^2 + (\ell-1)m - 1 - q^{-1}\textstyle\sum_{i=1}^{q}((i-1)(q+1) + \ell m + 1) \\ &\quad = n + q^2 - 2 - \tfrac{1}{2}(q^2-1) - m \\ &\quad = n + g - m - \tfrac{1}{2}(q-1) - 1 \end{aligned}$$

which is in $O(n)$ since $m > 2g - 2 \geq g - 1$. □

Remark. The complexities of Table 4.3 are very comparable with those of Table 3.4 on page 87, except in the case of the GJV where Power decoding suffers the $\nu = q$ penalty; therefore, it is important to know whether setting $\nu = q$ is really necessary. We demonstrated in Example 2.33 on page 29 that $\nu > 1$ can be useful for being very specific about which solutions one accepts in a 2D key equation, but this was a rather contrived example. Specifically, for Power decoding Hermitian codes, we could make a new 2D key equation with weights set instead to $\tilde{\nu} = 1$, $\tilde{\eta}_i = \lceil \eta_i/q \rceil$ and $\tilde{w}_j = \lfloor w_j/q \rfloor$. One can easily verify that the sought vector $(\Lambda_0, \ldots, \Lambda_{q-1})$ is still a solution to this new key equation; but due to integer rounding, we now possibly accept "false solutions" which would have been rejected with the more

Complexity of Power decoding Hermitian codes

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$q\ell^2 n^2$	$n^{7/3}\ell^2$
Alekhnovich	$\mathrm{M}(q\ell)\mathrm{P}(n)\log(n) + q^2\ell^2\mathrm{P}(n/q+q)$	$n^2\ell^3\log(n)^{2+o(1)}$
GJV	$\mathrm{M}(q\ell)\mathrm{P}(n)\log(n)^{O(1)}$	$n^2\ell^3\log(n)^{O(1)}$
D–D	$q^2\ell n\mathrm{P}(n/q+q)$	$n^{7/3}\ell\log^{1+o(1)}(n)$

Table 4.3: For relaxation, we have assumed $n \approx q^3$, i.e. that more or less all affine points were chosen. Remember that $\tilde{P}(n)$ for the D–D algorithm is $\mathrm{P}(n)$ since $G_j^{\mathrm{KE}}(x)$ are not powers of x.

precise weights. The question is whether this makes a practical difference: it seems it does. In the example $[343, 35, \geq 288]$ code of Example 4.25 with the $\nu = q$ weights, we could decode up to 173 errors almost always with $\ell = 2$. Using the above $\nu = 1$ weights, decoding fails in around 50% of the cases with only 170 errors, and for more errors it fails almost always. I have not analysed this case to get a theoretical bound to see if the difference is asymptotically noticeable, though. On the $\nu = 1$ weights, the GJV would have the relaxed asymptotic complexity $O(n^{5/3}\ell^3\log^{O(1)})$. ◆

Remark. It is intriguing that also Sudan decoding exhibits the strange behaviour of a *usual* decoding radius which is much higher than the worst case. Just counting linear equations versus variables in the Welch–Berlekamp case, i.e. Guruswami–Sudan with $s = \ell = 1$, yields the decoding radius $\frac{d-1}{2} - g$, see e.g. [BH08a]. However, similar to our remarks in the beginning of Section 4.3.1, then $Q(z) = \Lambda f - \Lambda z$ is a valid interpolation polynomial whenever $\tau < \frac{d-1}{2} - \frac{g}{2}$, which implies that some of these equations must be linearly dependent. A similar improvement can be done when $s = 1$ and $\ell > 1$ so we can Sudan decode up to $\tau < \frac{\ell}{\ell+1}n - \frac{1}{2}\ell m - \frac{g}{2}$ [Bee13]. I don't know of any work examining rigorously the probability that Guruswami–Sudan will succeed when more errors occur; however, Lee and O'Sullivan included some experimental results in [LO09] demonstrating that the Guruswami–Sudan usually decodes up to what seems to be $\frac{\ell}{\ell+1}n - \frac{1}{2}\ell m$. In most of their trials $s > 1$, so we can only compare the Welch–Berlekamp case with simple Gao decoding. Here, the only anomaly where Sudan decoding seems to behave differently than we expect key equation to is with the $[27, \geq 22, \geq 3]$ code over $\mathbb{F}_9$: they report 0 successes out of 10 000 trials of error weight 1; here, my experiments indicate that Gao decoding with $\ell = 1$ succeeds almost every time. ◆

4.5 Related work

As mentioned in Section 4.2, Power decoding was first described in [SSB06] as a "virtual extension" of a single GRS code into an interleaving of several such codes, and the algorithms those authors had developed for decoding Interleaved GRS

codes were then applied to single, low rate GRS codes [SSB09]. Similarly, the two new Power decoders discussed in this chapter can be straightforwardly turned into decoders for interleaved codes with these components codes. The greatest difference is in analysing the probability of failure: considering an interleaved code allows one to regard the errors in component codewords as independent except that they are non-zero on the same positions. For Power decoding, this is clearly not the case, so that case is more complex.

The algorithm Schmidt et al. proposed for solving the Power Syndrome key equation of Corollary 4.8 is the extension of Berlekamp–Massey for Multi-sequence LFSRs, which was also discussed in Section 2.7. It has complexity $O(\ell(n-k)^2)$, just as solving the 2D key equation using the Demand–Driven algorithm; most probably, the two algorithms could be shown to be step-wise equivalent on this type of 2D key equation.

Not all the connections between Power decoding and Sudan decoding described in this chapter are new; in particular, it was already noted in [SSB06] how the syndromes used in Power decoding coincide with those used in the Roth–Ruckenstein speedup, and how the decoding radii are close to each other. The authors also remarked that Power decoding and Coppersmith–Sudan both decode beyond half the minimum distance though neither are list decoders; however, they did not give the close algebraic connection given in Section 4.3.2.

Decoding Hermitian codes using a Syndrome key equation is rather classical, see e.g. [JLJ+89, JLJH92, SJH95]. These works treat more general AG codes than Hermitian codes. Using Sakata's generalisation of the Berlekamp–Massey algorithm, their decoding approach seem to have the same complexity as ours using the Demand–Driven algorithm (with $\ell = 1$). I am not aware of a D&C speedup of Sakata's algorithm; such a speedup is likely possible, though probably technically involved. My expectation is that such a D&C algorithm would resemble the Alekhnovich algorithm and achieve the same complexity.

Sabine Kampf was the first to describe Syndrome Power decoding of Hermitian codes, and she also focused only on one-point Hermitian codes [Kam11]. Only a trivial Gaussian elimination approach to solving the key equations was considered then, but in [KL13], together with Li she presented a Euclidean algorithm approach over $\mathbb{F}_{q^2}[x, y]$. This algorithm is only partially proved correct, though, and no complexity estimate is given. The latter might also be rather complicated: the algorithm is an extension of the Sugiyama-type algorithm for minimum-distance decoding Hermitian codes that Kampf had described in [Kam11], and even this algorithm was not given a full complexity analysis. Kampf and Li do not project the equations onto $\mathbb{F}_{q^2}[x]$ like we do, but it seems possible to argue about their algorithm using the module language from Chapter 2. In this light, it resembles a version of Mulders–Storjohann which starts with a matrix with only few rows and dynamically enlarges it whenever it can apply no more row operations; it does so by choosing a "worst" existing row $\boldsymbol{v}$ and adding the row $\Upsilon(y\Upsilon^{-1}(\boldsymbol{v}))$. Analysing the complexity of this using the

orthogonality defect and the related tools, we might be able to find the complexity. It would be interesting to understand why this dynamic approach, which at first glance might seem to tighter incorporate the algebraic rules of $\mathcal{R}$, does not yield better performance.

It is clearly interesting to see whether the Hermitian Gao key equation given here can be formulated for a wider class of AG codes. In particular, I foresee no problems whatsoever in extending the approach to one-point AG codes over "simple C_{ab} curves", which were those handled by Brander in his thesis [Bra10], since the important assumptions are all fulfilled there as well; in particular the curve has one point P_∞ at infinity with $\mathcal{R} = \mathcal{L}(\infty P_\infty) = \mathbb{F}_{q^2}[x, y]$, and the codes are evaluations of functions in $\mathcal{L}(mP_\infty)$. The related Miura–Kamiya curves investigated for decoding by Lee et al. [LBAO12] also seem to satisfy these properties. We also used in our derivation that G was univariate in x, but that does not seem to be a crucial property, which would also be likely to not hold for more general codes.

The uncertainty gap between $\frac{d-1}{2} - \frac{g}{2}$ and $\frac{d-1}{2}$, where decoding *should* always be possible, but the key equation approach will sometimes fail, can be closed when using Syndrome key equation with Feng et al.'s majority voting scheme [FWRT94]. This scheme can also be applied in conjunction with Sakata's algorithm for solving the key equation [SJH95]. It would be highly interesting to examine majority voting for the Gao key equation, to understand what it *means* in the module minimisation approach, and whether it can easily be applied as an extension to any or all of the minimisation algorithms. Lee et al. [LBAO12] showed how to use a Gröbner basis approach akin to both Guruswami–Sudan with $s = \ell = 1$ as well as majority voting to decode up to $\frac{d-1}{2}$, but I have not yet examined how related it is.

I have not yet looked deeper into the equivalence between Power Syndrome and Power Gao decoding in the Hermitian code case, but it is clearly interesting to see whether the arguments from GRS codes extend; I am rather optimistic in this regard. Similarly, I have not extended Proposition 4.7 on page 101 and verified that the failure behaviour is invariant of the sent codeword, but the argument seems to generalise. It is probably more involved, however, to generalise the orthogonalities between Sudan decoding and Power decoding; a useful description in this regard is the Welch–Berlekamp type description of decoding Hermitian codes in [JH04], as well as the Q-interpolation by key equations (analogous to in Section 3.4) in [BH08b].

In [LSN13], Li, Sidorenko and I developed a Power decoder for Chinese Remainder codes, and we also analysed the failure probability for the interleaved case. Chinese Remainder codes are the $\mathbb{Z}$-analogue of GRS codes, and the setup we use is completely an analogue of the module approach presented here. To perform "module minimisation", or "lattice basis reduction" as it is customarily called in the number theory field, we use the famous Lenstra–Lenstra–Lovász algorithm [LLL82]. Analysis is complicated by the fact that this algorithm does not promise the *shortest* solution, but rather one which is *close* to the shortest; finding the shortest vector in the row space of a $\mathbb{Z}$ matrix is known to be NP-hard [Ajt98]. The decoding algorithm still

works very well, however. An interesting facet of this work is that the Berlekamp–Massey algorithm does not have a $\mathbb{Z}$ analogue, which is why the Berlekamp–Massey type algorithms for solving the "powered" key equations for GRS codes can not be accommodated to Chinese Remainder codes; one really needs the module language as a generalisation of the Euclidean algorithm.

Chapter 5

Wu list decoding

Wu proposed in [Wu08] a completely new list decoder for GRS codes. It was presented as an analytic extension of the Berlekamp–Massey algorithm applied to the classical key equation involving the error locator and the syndrome polynomial. Wu observed that when more errors than half the minimum distance occurs, the error locator is a small $\mathbb{F}[x]$-linear combination of the two polynomials computed by the Berlekamp–Massey. By utilising the knowledge that the error locator has zeroes at the error positions, Wu described how to find this linear combination using a generalisation of the Guruswami–Sudan algorithm. Perhaps surprisingly, the resulting decoding algorithm has exactly the same decoding radius as the Guruswami–Sudan.

More generally, one can see the paradigm of Wu's method as that of solving a simple key equation, i.e. $\gamma S \equiv \delta \mod G$, whenever one has certain side-information on the evaluations of γ and δ. We will demonstrate this paradigm twice, in the original setting for GRS codes in Section 5.2, and a new application for binary Goppa codes in Section 5.3; the nature of the side-information is going to be different in these two cases.

The host of observations that Wu needed from the results of the Berlekamp–Massey algorithm, more verbosely described and proved in [Nie10], turn out all to follow from the fact that they constitute "half" of a Gröbner basis of the module for the key equation, in the sense of Section 2.5. Therefore, applying the general results of Chapter 2 makes us capable of deriving this part of the algorithm very easily.

The generalisation of the Guruswami–Sudan algorithm for solving the "rational interpolation" can be completely detached from the decoding problem, and we begin by describing this in Section 5.1, as well as give algorithms for solving it fast.

Since this is the computationally heavy part of the algorithm, the complexity of the decoding algorithms will follow immediately.

Contributions

- Description of Wu's paradigm for solving key equations by drawing on the Gröbner basis language; however, see also Section 5.4. This appeared in an equivalent form in [BHNW13].
- Closed expressions for good choices of parameters s and ℓ for rational interpolation by pointing out a general duality between the parameter choices of rational interpolation and the Guruswami–Sudan.
- Generalised module approach for fast Q-interpolation for rational interpolation. This appeared in essence in [BHNW13].
- The root-finding method we describe is in essence that suggested by Wu in [Wu08], though we have extended it to handle the case $x \mid p_2(x)$, and we point out the use of the fast D&C speedup of Roth–Ruckenstein described by Alekhnovich, see Proposition 3.6 on page 50.
- Proposition 5.22 describing (another) duality in the parameter choices between the Wu list decoder and the Guruswami–Sudan when decoding GRS codes; we discovered this but did not write it in a preprint of [BHNW13], and one of the anonymous reviewers independently discovered it as well and kindly pointed it out to us.
- Applied the Wu list decoding method to binary Goppa codes. This appeared in [BHNW13].
- Parallel decoding variant of Wu decoding binary Goppa codes, Section 5.3.1. This appeared as a remark in [BHNW13] but has been greatly expanded for the thesis.
- Proposition 5.33 describing (yet another) duality in the parameter choices between the Wu list decoder for binary Goppa codes and the Guruswami–Sudan with Kötter–Vardy multiplicity assignment for binary Alternant codes. This appeared in [BHNW13] after having been pointed out to us by the same anonymous reviewer as before. Using the closed expressions for choices of parameters in Wu list decoding, we therefore get parameter choices for this Guruswami–Sudan decoding of Alternant codes. This is to my knowledge the first such optimised closed form parameter choices, even for this special case of the decoding algorithm.

5.1 Rational interpolation

We will first describe a solution to the problem of finding "small" rational curves that go through at least some number of prescribed points; this will turn out to be the main computational burden in Wu's list decoder. Notice that the analogous problem of finding low-degree polynomials that go through some of a list of prescribed points

is exactly Reed–Solomon decoding, and indeed, the rational interpolation method we present here is a generalisation of the Guruswami–Sudan algorithm. It was first described by Wu [Wu08], but we will use a different formulation using partially homogenised polynomials, originally introduced by Trifonov [Tri10a].

We are basically interested in a rational expression $\frac{p_1}{p_2} \in \mathbb{F}(x)$ with numerator and denominator of low degrees, which goes through at least some τ out of n points $\big((x_0, \beta_0), \ldots, (x_{n-1}, \beta_{n-1})\big)$ where all $x_i \in \mathbb{F}$ while $\beta_i \in \mathbb{F} \cup \{\infty\}$. To handle the points at infinity, we can instead consider these as partially projective points $(x_i, y_i : z_i) \in \mathbb{F} \times \mathbb{P}^1(\mathbb{F})$ with $\frac{y_i}{z_i} = \beta_i$ whenever $\beta_i \neq \infty$ and $(y_i, z_i) = (1, 0)$ otherwise. In this language, the interpolation amounts to finding low-degree polynomials $p_1, p_2 \in \mathbb{F}[x]$ such that for at least τ values of i, we have $z_i p_1(x_i) - y_i p_2(x_i) = 0$.

Introduce first the structure $\mathbb{F}[x][y : z]^\ell$ by which we mean trivariate polynomials which are homogeneous of total degree ℓ in y and z; i.e a polynomial which can be written $P = \sum_{i=0}^{\ell} p_i(x) y^i z^{\ell - i}$; when the variables are important, we employ the notation $P(x, y : z)$. Notice that $\mathbb{F}[x][y : z]^\ell$ is an $\mathbb{F}[x]$-module, but that multiplying e.g $P_1 \in \mathbb{F}[x][y : z]^{\ell_1}$ with $P_2 \in \mathbb{F}[x][y : z]^{\ell_2}$ yields an element in $\mathbb{F}[x][y : z]^{\ell_1 + \ell_2}$. We can directly carry over many definitions and results from $\mathbb{F}[x, y, z]$, such as "zero of multiplicity" from Definition 3.1 on page 48. To get better acquainted with this definition for elements of $\mathbb{F}[x][y : z]^\ell$, we first prove two small lemmas:

Lemma 5.1. *$P(x, y : z) = \sum_{t=0}^{\ell} P_t(x) y^t z^{\ell - t} \in \mathbb{F}[x][y : z]^\ell$ has a root $(x_0, y_0 : z_0)$ of multiplicity s if and only if $\overline{P} = \sum_{t=0}^{\ell} P_{\ell - t}(x) y^t z^{\ell - t}$ has a root $(x_0, z_0 : y_0)$ of multiplicity s.*

Proof. Since $P(x, y : z) = \overline{P}(x, z : y)$ then clearly $P(x + x_0, y + y_0 : z + z_0) = \overline{P}(x + x_0, z + z_0 : y + y_0)$, so the statement trivially follows from the definition of root of multiplicity. □

Lemma 5.2. *$P(x, y : z) = \sum_{t=0}^{\ell} P_t(x) y^t z^{\ell - t} \in \mathbb{F}[x][y : z]^\ell$ has a root $(x_0, y_0 : z_0)$ of multiplicity s if and only if there exist $\grave{P}_h(x, y : z) \in \mathbb{F}[x][y : z]^{\ell - h}$ such that*

$$P(x, y : z) = \sum_{j + h \geq s} (x - x_0)^j (z_0 y - y_0 z)^h \grave{P}_h(x, y : z)$$

Proof. Clearly if $P(x, y : z)$ can be written in such a form, then $P(x + x_0, y + y_0 : z + z_0)$ has no monomials of degree less than s, so by definition $P(x, y : z)$ has a zero of multiplicity s.

For the other direction, assume first that $z_0 \neq 0$. Since for any $\gamma \in \mathbb{F}^*$, then $0 = P(x_0, y_0 : z_0) = \gamma^{-\ell} P(x_0, \gamma y_0 : \gamma z_0)$, we can assume that $z_0 = 1$. Since $P(x + x_0, y + y_0 : z + 1) = \sum_{t=0}^{\ell} P_t(x + x_0)(y + y_0)^t (z + 1)^{\ell - t}$ has no monomials of degree less than s, then in particular this means when setting $z = 0$ that $\sum_{t=0}^{\ell} P_t(x + x_0)(y + y_0)^t$ has no monomials of degree less than s, so there must exist $q_{jh} \in \mathbb{F}$ such that

$$\sum_{t=0}^{\ell} P_t(x + x_0)(y + y_0)^t = \sum_{j + h \geq s} q_{jh} x^j y^h \iff \sum_{t=0}^{\ell} P_t(x) y^t = \sum_{j + h \geq s} q_{jh} (x - x_0)^j (y - y_0)^h$$

But then

$$P(x,y:z) = z^\ell \sum_{t=0}^{\ell} P_t(x)(\tfrac{y}{z})^t = z^\ell \sum_{j+h\geq s} q_{jh}(x-x_0)^j(\tfrac{y}{z}-y_0)^h$$

which immediately gives the sought since $z_0 = 1$.

If on the other hand $z_0 = 0$ then $y_0 \neq 0$. Since $P(x,y:z)$ has root $(x_0,y_0:0)$ of multiplicity s, then $\overline{P}(x,y:z)$ has root $(x_0,0:y_0)$ of multiplicity s by Lemma 5.1. By the already proved part of this lemma, then we can find $\grave{P}_t$ such that $\overline{P}(x,y:z) = \sum_{j+h\geq s}(x-x_0)^j(y_0y-z_0z)^h\grave{P}_h(x,y:z)$. But that means $P(x,y:z) = \sum_{j+h\geq s}(x-x_0)^j(z_0y-y_0z)^h\grave{P}_h(x,z:y)$. □

Corollary 5.3. *$P \in \mathbb{F}[x][y:z]^\ell$ has a zero $(x_0,y_0:z_0)$ of multiplicity s with $z_0 \neq 0$ if and only if $P(x,y:1) \in \mathbb{F}[x,y]$ has a zero $(x_0,y_0/z_0)$ of multiplicity s.*

Now for the theorem leading to the rational interpolation method; it's basically just a projective extension of Theorem 3.2 on page 48. Note that we allow non-integral weights θ_1,θ_2, which will, curiously, be necessary when list decoding binary Goppa codes in Section 5.3.

Theorem 5.4. *Let ℓ, s and τ be positive integers, and let $(x_i,y_i:z_i) \in \mathbb{F}\times\mathbb{P}^1(\mathbb{F})$ for $i = 1,\ldots,n$ be n partially projective points. Assume that $Q(x,y:z) \in \mathbb{F}[x][y:z]^\ell$ is non-zero and such that $(x_i,y_i:z_i)$ are zeroes of multiplicity s for all $i = 1,\ldots,n$, and $\deg_{1,\theta_1,\theta_2} Q < s\tau$, for $\theta_1,\theta_2 \in \mathbb{R}_+ \cup \{0\}$. Any two coprime polynomials p_1,p_2 satisfying $\deg p_1 \leq \theta_1$, $\deg p_2 \leq \theta_2$, as well as $z_ip_1(x_i) - y_ip_2(x_i) = 0$ for at least τ values of i, will satisfy $Q(x,p_1(x):p_2(x)) = 0$.*

Proof. Let $\hat{Q}(x) = Q(x,p_1(x):p_2(x))$. Consider an i such that $z_ip_1(x_i) - y_ip_2(x_i) = 0$. By Lemma 5.2, we have $Q(x,y:z) = \sum_{j+h\geq s}(x-x_i)^j(z_iy-y_iz)^h\grave{Q}_h(x,p_1(x):p_2(x))$ for some $\grave{Q}_h \in \mathbb{F}[x][y:z]^\ell$, and so

$$\hat{Q}(x) = \sum_{j+h\geq s}(x-x_i)^j(z_ip_1(x)-y_ip_2(x))^h\grave{Q}_h(x,y:z)$$

Since $z_ip_1(x)-y_ip_2(x)$ has a zero at x_i then $(x-x_i) \mid (z_ip_1(x)-y_ip_2(x))$. Therefore, $(x-x_i)^s$ must divide each term in the above sum and therefore $\hat{Q}$. This is true for all τ of the choices of i where $z_ip_1(x_i)-y_ip_2(x_i) = 0$, which means $\hat{Q}$ is divisible by a polynomial of degree $s\tau$. However, $\deg\hat{Q} = \deg\big(Q(x,p_1(x):p_2(x))\big) \leq \deg_{1,\theta_1,\theta_2} Q < s\tau$, since $\deg p_1 \leq \theta_1$ and $\deg p_2 \leq \theta_2$. Therefore $\hat{Q} = 0$. □

Just as with Theorem 3.2, we need to analyse the interplay between the problem parameters n,θ_1 and θ_2, the decoding radius τ, and the parameters s and ℓ to determine when we can be sure that such a Q can be found. We can mimic the first part easily:

Definition 5.5. Let the *Wu satisfiability function* $E_{\text{Wu}}^{[n,\theta]}(s,\ell,\tau)$ be given by

$$E_{\text{Wu}}^{[n,\theta]}(s,\ell,\tau) = (\ell+1)s\tau - \tbinom{\ell+1}{2}\theta - \tbinom{s+1}{2}n$$

Lemma 5.6. *In the context of Theorem 5.4, a Q exists if $E_{\text{Wu}}^{[n,\theta_1+\theta_2]}(s,\ell,\tau) > 0$.*

Proof. This is completely analogous to Proposition 3.4 on page 49: the number of variables is the number of coefficients at our disposal, i.e.

$$\begin{aligned}\sum_{i=0}^{\ell}\left(s\tau - \lfloor i\theta_1 + (\ell-i)\theta_2\rfloor\right) &\geq \sum_{i=0}^{\ell}\left(s\tau - i\theta_1 - (\ell-i)\theta_2\right)\\ &= (\ell+1)s\tau - \tbinom{\ell+1}{2}(\theta_1+\theta_2)\end{aligned}$$

For the number of constraints, one uses Corollary 5.3 to see that each point $(x_i, y_i : z_i)$ with $z_i \neq 0$, specifies $\binom{s+1}{2}$ linear constraints on the coefficients of $Q(x, y : 1)$, i.e. the coefficients of Q; when $z_i = 0$ one regards $Q(x, 1 : y)$ instead. □

As we will see later, in decoding applications the value $\theta = \theta_1 + \theta_2$ will depend on the code parameters and τ in various ways, so though τ of Theorem 5.4 will correspond to the decoding radius, we can not already now give a sensible bound on it. However, treating θ as an independent variable, we can by reusing analysis of Chapter 3 easily arrive at the following, which within bounds for τ, n and θ gives good choices of s and ℓ. Due to a technicality later in the decoders, we also need to be sure that we can find parameters satisfying $\ell/s \geq \tau/\theta$.

Proposition 5.7. *If $\tau^2 > n\theta$, then $E_{\text{Wu}}^{[n,\theta]}(s,\ell,\tau) > 0$ if we choose ℓ and s as:*

$$s = \lfloor s_{\min} + 1\rfloor \qquad \ell = \begin{cases} \lfloor \frac{n}{\tau}\rfloor & \text{if } \theta = 0 \\ \left\lfloor \frac{\tau}{\theta}s + \frac{1}{2} - \frac{\sqrt{D(s)}}{\theta}\right\rfloor & \text{if } \theta > 0 \end{cases}$$

where

$$s_{\min} = \frac{\theta(n-\tau)}{\tau^2 - n\theta} \qquad D(s) = (s - s_{\min})(\tau^2 - n\theta)s + \tfrac{\theta^2}{4}$$

Furthermore, if $\theta > 0$ then there exists an integer $\delta \leq \frac{2\theta}{n-\tau} \in O(n)$ such that $E_{\text{Wu}}^{[n,\theta]}(s+\delta,\ \lceil\frac{\tau}{\theta}(s+\delta)\rceil,\ \tau) > 0$ for the above choice of s.

Proof. By insertion we see that $E_{\text{Wu}}^{[n,\theta]}(s,\ell,\tau) = E_{\text{GS}}^{[n,\theta+1]}(s,\ell,n-\tau)$, so we wish to reuse the parameter choices given by Proposition 3.11 on page 53. Introduce $k_{\text{GS}} = \theta+1$ and $\tau_{\text{GS}} = n-\tau$; the proof of the proposition assumed certain properties which follow from being legal decoding parameters, in particular $n > \tau_{\text{GS}}$ and $n - \tau_{\text{GS}} \geq k_{\text{GS}} - 1$. The first is obviously true and the latter follow from $\tau^2 > n\theta$. Note also that nowhere was it assumed that k is an integer. We can then use the proposition as long as we satisfy its two main assumptions: $k_{\text{GS}} > 1$ and $\tau_{\text{GS}} > n - \sqrt{n(n-(n-k_{\text{GS}}+1))}$.

Assume first $\theta > 0$ whence $k_{\text{GS}} > 1$. Since we have assumed $\tau^2 > n\theta$ then $(n - \tau_{\text{GS}})^2 > n(k_{\text{GS}}-1)$ which is true if and only if $\tau_{\text{GS}} < n - \sqrt{n(n-k_{\text{GS}}+1)}$. Therefore we can use the parameters of Proposition 3.11, replacing $k_\circ$ with θ, τ with $n-\tau$ and $\bar{\tau}$ with τ.

If on the other hand $\theta = 0$, we cannot use Proposition 3.11. But then we need to satisfy $E_{\mathrm{Wu}}^{[n,0]}(s,\ell,\tau) = (\ell+1)s\tau - \binom{s+1}{2}n > 0$. Setting $s = 1$ as the minimal possible, we immediately get $\ell = \lfloor \frac{n}{\tau} \rfloor$. Note that since $s_{\min} = \frac{\theta(n-\tau)}{\tau^2 - n\theta} = 0$ then $s = 1 = \lfloor s_{\min} + 1 \rfloor$.

For the second statement, assume again $\theta > 0$. We need to dig deeper into the proof of Proposition 3.11: paraphrasing, then it was actually shown that for any value of $s > s_{\min}$ then choosing ℓ in the range $\left] \frac{\tau}{\theta}s - \frac{1}{2} - \frac{\sqrt{D(s)}}{\theta}; \frac{\tau}{\theta}s - \frac{1}{2} + \frac{\sqrt{D(s)}}{\theta} \right[$ yields parameters satisfying $E_{\mathrm{GS}}^{[n,k]}(s,\ell,\tau_{\mathrm{GS}}) > 0$. This range clearly holds an integer $\tilde{\ell} \geq \frac{\tau}{\theta}s$ at least when $\frac{\sqrt{D(s)}}{\theta} \geq \frac{3}{2}$. That restriction becomes

$$\begin{aligned} (s - s_{\min})(\tau^2 - n\theta)s + \tfrac{\theta^2}{4} &\geq \tfrac{9}{4}\theta^2 \qquad \Longleftrightarrow \\ s - s_{\min} &\geq \tfrac{2\theta^2}{\tau^2 - n\theta} s^{-1} \end{aligned}$$

Upper bounding s^{-1} by $s_{\min}^{-1}$ and simplifying we get that there is a satisfactory s with $s - s_{\min} \leq \frac{2\theta}{n-\tau} \in O(n)$. □

Similarly, reusing the analyses of Section 3.1, one can replicate most of the results on the asymptotic behaviour of the parameter choices. Note that the poor bound on the mentioned δ reflects reality quite badly: firstly, usually $\frac{\theta}{n-\tau}$ is usually in $O(1)$, so expecting $O(n)$ is very pessimistic. Secondly, the result is necessary for technically securing parameters satisfying inequalities such as $\ell \geq s$ while knowing $\tau \geq \theta$ (in e.g. Algorithm 6 on page 143); but for almost *all* code parameters the closed form parameter choices will satisfy the necessary inequalities directly.

Though Theorem 5.4 is a parallel to Theorem 3.2, it will not be quite enough for our applications to decoding: when we later need to solve a rational interpolation problem for decoding, we seek p_1 and p_2 which interpolate on the error positions, and therefore an unknown number of points, but their maximal degrees increase with the number of points they interpolate. This means that we can't use Theorem 5.4 directly: setting τ low while the allowed degrees of p_1, p_2 high would not allow us to construct Q, while setting τ high would not guarantee that we would find p_1 and p_2 when only few points were interpolated. Luckily, we have the following lemma which says that the Q we construct for high τ will also find p_1 and p_2 that interpolate fewer points, as long as their degrees decrease appropriately:

Lemma 5.8. *Let $Q(x, y : z)$ satisfy the requirements of Theorem 5.4 for some choice of $\tau, \ell, s, \theta_1, \theta_2$. Then Q also satisfies the requirements for $\tilde{\tau}, \ell, s, \tilde{\theta}_1, \tilde{\theta}_2$ as long as*

$$\min\{\theta_1 - \tilde{\theta}_1 \ , \ \theta_2 - \tilde{\theta}_2\} \geq \tfrac{s}{\ell}(\tau - \tilde{\tau})$$

Proof. Since the interpolation points and multiplicity as well as the list size have not changed, we only need to show $\deg_{1,\tilde{\theta}_1,\tilde{\theta}_2} Q < s\tilde{\tau}$. We have:

$$\begin{aligned} \deg_{1,\tilde{\theta}_1,\tilde{\theta}_2} Q &\leq \deg_{1,\theta_1,\theta_2} Q - \min\{i(\theta_2 - \tilde{\theta}_2) + (\ell - i)(\theta_1 - \tilde{\theta}_1) \mid 0 \leq i \leq \ell\} \\ &< s\tau - \ell \min\{\theta_1 - \tilde{\theta}_1,\ \theta_2 - \tilde{\theta}_2\} \end{aligned}$$

Therefore Q satisfies the degree constraints whenever

$$s\tau - \ell \min\{\theta_1 - \tilde{\theta}_1,\ \theta_2 - \tilde{\theta}_2\} \leq s\tilde{\tau} \qquad \Longleftrightarrow$$
$$\min\{\theta_1 - \tilde{\theta}_1,\ \theta_2 - \tilde{\theta}_2\} \geq \frac{s}{\ell}(\tau - \tilde{\tau}) \qquad \square$$

5.1.1 Root-finding for rational interpolation

After having computed the interpolation polynomial Q, one needs to find the y and z roots p_1, p_2 predicted by Theorem 5.4. First note that for any p_1, p_2 with $p_2 \neq 0$ such that $Q(x, p_1(x) : p_2(x)) = 0$ then also $Q(x, \frac{p_1(x)}{p_2(x)} : 1) = 0$. Therefore, it suffices to find $\mathbb{F}(x)$ y-roots of $\tilde{Q}(x, y) = Q(x, y : 1) \in \mathbb{F}[x][y]$. Wu[1] and Siegel [WS01] generalised the Roth–Ruckenstein root-finding algorithm [RR00]—which we already mentioned in Proposition 3.6 on page 50—to work for AG codes over curves, and rational expression root-finding is a special case of this. In fact, in this case the root-finding algorithm becomes identical to that of [RR00], except that one considers any found polynomial $p(x)$ as an approximate power series expansions of a root $\frac{p_1}{p_2}$, i.e. $p(x) \equiv \frac{p_1(x)}{p_2(x)} \mod x^m$ for any desired m. As long as m is chosen large enough compared to the degrees of the sought p_1 and p_2, one can use Padé approximation techniques to retrieve them. This leads to:

Proposition 5.9. *Given $Q \in \mathbb{F}[x][y : z]^\ell$ with $\deg_x Q \leq N$, there exists an algorithm for finding all coprime $p_1, p_2 \in \mathbb{F}[x]$ with $\max\{\deg p_1, \deg p_2\} < N/2$ and such that $Q(x, p_1(x) : p_2(x)) = 0$ in complexity $O(\ell^2 \mathrm{P}(N) \log N)$, assuming that $\ell, q \in O(N)$ where q is the cardinality of $\mathbb{F}$.*

Proof. [sketch] If $p_2 = 0$ then $p_1(x) = 1$ by coprimeness. So we simply check whether $Q(x, 1 : 0) = 0$. For the remaining cases, it suffices if we in the described speed can find all $\frac{p_1}{p_2} \in \mathbb{F}(x)$ such that $\tilde{Q}(x, \frac{p_1(x)}{p_2(x)}) = 0$, where $\tilde{Q} = Q(x, y : 1)$, and where $\max\{\deg p_1, \deg p_2\} < N/2$.

Assume first $x \nmid p_2(x)$. Then there is a polynomial $\hat{p} \in \mathbb{F}[x]$ such that $\hat{p} \equiv \frac{p_1}{p_2} \mod x^N$; clearly $\tilde{Q}(x, \hat{p}) \equiv 0 \mod x^N$. Following the remark on page 51, the Roth–Ruckenstein root-finding method [RR00] with the D&C speedup by Alekhnovich actually finds all $p \in \mathbb{F}[x]$ such that $\tilde{Q}(x, p(x)) \equiv 0 \mod x^N$; thus, $\hat{p}$ must be among these found polynomials. Roth and Ruckenstein proves that this list will contain at most $\deg_y \tilde{Q} \leq \ell$ elements.

Now, $\hat{p} \equiv \frac{p_1}{p_2}$ is the same as $p_2 p \equiv p_1 \mod x^N$. This is a simple key equation. Following Section 2.5, we minimise the matrix $\begin{pmatrix} 1 & \hat{p} \\ 0 & x^N \end{pmatrix}$ to get $\begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \end{pmatrix}$ in weak Popov form, and number it so that $\mathrm{LP}(\boldsymbol{g}_1) = 1$ so $\mathrm{LP}(\boldsymbol{g}_2) = 2$. We must have $\deg \boldsymbol{g}_1 + \deg \boldsymbol{g}_2 = N$ since the determinant is unchanged by minimisation. Thus only either $\boldsymbol{g}_1$ or $\boldsymbol{g}_2$ has degree less than $N/2$, say $\boldsymbol{g}_i$. The only possibility is now

[1] Xin-Wen Wu and not Yinquan Wu, originator of the Wu list-decoder.

$(p_2, p_1) = \boldsymbol{g}_i$ or there is no solution satisfying our requirements: any other solution to the key equation must be an $\mathbb{F}[x]$-linear combination of $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$; but it will have degree greater than $N/2$ if $\boldsymbol{g}_{3-i}$ enters in the combination (since $\boldsymbol{g}_1, \boldsymbol{g}_2$ form a Gröbner basis with respect to $\preceq_{1,\mathbf{0}}$), and non-coprime elements if it is not just an $\mathbb{F}$-scaling of $\boldsymbol{g}_i$. Now, if $\gcd(g_{i,1}, g_{i,2}) = q(x) \neq 1$, then $q \mid \det \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \end{pmatrix} = x^N$, so $q(x) = x^t$ for some t. But we assumed that $x \nmid p_2$, so either $q(x) = 1$, in which case $(p_2, p_1) = \boldsymbol{g}_i$ is a valid solution, or $q(x) \neq 1$, in which the polynomial p did not yield a valid rational root of $\tilde{Q}$.

The case $x \mid p_2$ is now easy. First note that $x \nmid p_1$. Analogous to before, $Q(x, p_1 : p_2) = 0$ implies $\bar{Q}(x, \frac{p_2}{p_1}) = 0$, where $\bar{Q}(x, z) = Q(x, 1 : z)$. Therefore, in the manner above, we just find all $\mathbb{F}(x)$-roots $\frac{p_2}{p_1}$ of $\bar{Q}$ where $x \nmid p_1$. Together with the roots found before, we have all roots.

By Proposition 3.6, it costs $O(\ell^2 \mathrm{P}(N) \log N)$ to call to Roth–Ruckenstein D&C root-finding, while each of the up to ℓ module minimisations costs $O(\mathrm{P}(N) \log N)$ if we use the Alekhnovich algorithm, see Table 2.4. Doing this twice is obviously in the same complexity. □

Corollary 5.10. *In the context of Theorem 5.4, if $\tau^2 > n(\theta_1 + \theta_2)$ we can find all p_1, p_2 such that $Q(x, p_1(x) : p_2(x)) = 0$ and satisfying $\deg p_1 \leq \theta_1$ and $\deg p_2 \leq \theta_2$ in complexity $O(\ell^2 \mathrm{P}(s\tau) \log(s\tau))$, assuming $q \in O(sn)$.*

Proof. Taking $N = 2s\tau = 2 \deg_x Q$, then from Proposition 5.9, we can find p_1 and p_2 in the specified complexity as long as $\theta_1, \theta_2 < s\tau$. But if $\tau^2 > n(\theta_1 + \theta_2)$ then $\theta_1 + \theta_2 \leq \tau$ since $\tau \leq n$. □

Remark. There is an easy check which will often allow one to skip one of the two root-finding-attempts in the method outlined in Lemma 5.8: assume that $p_1 : p_2$ is a root of $Q(x, y : z)$ with $x \mid p_2$. Then

$$0 = Q(x, p_1 : p_2) = Q_\ell(x) p_1^\ell(x) + p_2(x) \sum_{i=0}^{\ell-1} Q_i(x) p_1^i(x) p_2(x)^{\ell-i-1}$$

which implies $p_2 \mid Q_\ell(x)$ since $\gcd(p_1, p_2) = 1$. Therefore, if $x \nmid Q_\ell$, there can be no rational roots of $\tilde{Q}$ with x dividing the denominator, so there is no need to perform the root-finding attempt on $\bar{Q}$. On the other hand, if there is, one can check if $x \mid Q_0$; if not, then one can instead skip the root-finding in $\tilde{Q}$ and apply only that in $\bar{Q}$.

One can take this observation a step further to find all rational roots completely without the Padé approximation: since $p_2 \mid Q_\ell(x)$, we can do a "Gauss' Lemma"-like change of variables in $\tilde{Q}$ to end up with a new polynomial with roots only in $\mathbb{F}[x]$: consider $\hat{Q}(x, y) = Q_\ell^{\ell-1}(x) \tilde{Q}(x, \frac{y}{Q_\ell(x)})$. Clearly $\hat{Q} \in \mathbb{F}[x, y]$ and has $\hat{Q}_{[\ell]} = 1$. By the argument just above, any $\frac{p_1'}{p_2'} \in \mathbb{F}(x)$ such that $\hat{Q}(x, \frac{p_1'}{p_2'}) = 0$ must have $p_2' = 1$. Note also that if $\frac{p_1}{p_2} \in \mathbb{F}(x)$ are such that $\tilde{Q}(x, \frac{p_1}{p_2}) = 0$, then

$$\hat{Q}(x, \tfrac{Q_\ell}{p_2} p_1) = Q_\ell^{\ell-1} \tilde{Q}(x, \tfrac{p_1}{p_2}) = 0$$

Thus the $\mathbb{F}(x)$-roots of $\tilde{Q}$ are represented as $\mathbb{F}[x]$-roots of $\hat{Q}$. We can therefore just apply Roth–Ruckenstein's root-finding algorithm to $\hat{Q}$, possibly with the D&C speedup, and then easily map the found roots back to those of $\tilde{Q}$.

Though simpler and arguably more elegant, it is quite likely, however, that this strategy is slower than that of Proposition 5.9, since $\deg_x \hat{Q} \in O(\ell N)$. ◆

5.1.2 Finding Q in an explicit module

We will now show how the techniques described in Section 3.2 applies to the construction of a satisfactory Q polynomial in the rational interpolation case. The projective points complicate certain matters somewhat, but the general idea is completely unchanged.

We will again assume that $s \leq \ell$. However, to use Proposition 5.7, we will be restricting ourselves to cases where $\tau^2 > n\theta$, and in that case $\tau/\theta > 1$ and we can choose the parameters with Proposition 5.7 satisfying $s \leq \ell$.

Definition 5.11. Let $\mathcal{W}_{s,\ell} \subset \mathbb{F}[x][y:z]^\ell$ denote the space of all polynomials passing through all the points $(x_1, y_1 : z_1), \ldots, (x_n, y_n : z_n)$ with multiplicity s.

Our goal is then to find a non-zero $Q \in \mathcal{W}_{s,\ell}$ of lowest possible $(1, \theta_1, \theta_2)$-weighted degree. Just like $\mathcal{M}_{s,\ell}$ of Definition 3.21 on page 60, $\mathcal{W}_{s,\ell}$ is an $\mathbb{F}[x]$-module. The approach is again first to give an explicit basis for $\mathcal{W}_{s,\ell}$, and then use the techniques of Chapter 2 to find a Gröbner basis of an appropriate monomial ordering.

Let us assume without loss of generality that each $z_i \in \{0, 1\}$. Let $L = \{i \mid z_i = 0\}$ and $\overline{L} = \{i \mid z_i \neq 0\}$. Introduce now a number of polynomials, generalising what we needed for Guruswami–Sudan interpolation in Definition 3.22 on page 60:

$$R_y(x) \colon \forall i \,.\, R_y(x_i) = y_i \qquad G(x) = \prod_{i=1}^{n} (x - x_i)$$
$$R_z(x) \colon \forall i \,.\, R_z(x_i) = z_i \qquad g_z(x) = \prod_{i \in L} (x - x_i) = \gcd(G, R_z)$$

Also $\deg R_y < n$ and $\deg R_z < n$, making them both unique. Furthermore, by the definition of g_z, there must exist $q_1, q_2 \in \mathbb{F}[x]$ such that $g_z(x) = q_1(x)G(x) + q_2(x)R_z(x)$. Let then $h_y(x) = \big(q_2(x)R_y(x) \bmod G(x)\big)$. Note that $h_y(x_i) = q_2(x_i)y_i$ for all $i = 1, \ldots, n$.

Extend the notation $Q_{[t]}(x)$ to elements of $\mathbb{F}[x][y:z]^\ell$ such that $Q_{[t]}(x)$ is the $y^t z^{\ell-t}$-coefficient of $Q(x, y:z)$. We begin with a small lemma, similar to Lemma 3.23 on page 60:

Lemma 5.12. *Let $Q \in \mathcal{W}_{s,\ell}$ with $\deg_y Q = t$. If $t < s$ then $(\frac{G}{g_z})^{s-t} \mid Q_{[t]}(x)$. If $t > \ell - s$ then $g_z(x)^{t-(\ell-s)} \mid Q_{[t]}(x)$.*

Proof. Consider first $t < s$. By Corollary 5.3, for any i such that $z_i \neq 0$ then Q interpolates $(x_i, y_i : z_i)$ if and only if $\tilde{Q}(x, y) = Q(x, y : 1)$ interpolates $(x_i, y_i/z_i)$ with multiplicity s. Since $\deg_y \tilde{Q} = \deg_y Q = t$, then having this zero implies that $(x - x_i)^{s-t}$ divides $\tilde{Q}_{[t]}(x) = Q_{[t]}(x)$, by the proof of Lemma 3.23 on page 60. Collecting for each such i gives the sought.

For $t > \ell - s$, we need to slightly improve the approach of Lemma 3.23. Choose first $i \in L$ so $z_i = 0$. Therefore $Q(x + x_i, y + y_i : z + z_i) = \sum_{j=0}^{t} Q_{[j]}(x + x_i)(y + y_i)^j z^{\ell-j}$. We see that choosing the tth term and multiplying out, $Q_{[t]}(x + x_i)y_i^t y^0 z^{\ell-t}$ is the only term with yz-degree $y^0 z^{\ell-t}$. This can have no monomials of degree less than s, so if $\ell - t < s$ then $x^{s-(\ell-t)} \mid Q_{[t]}(x + x_i)$. This implies the claim similar to before. □

Now for the basis. Recall again the positive function $\mathrm{pos}(\cdot)$, see e.g. Appendix A. Note the easy identity $\mathrm{pos}(x) - \mathrm{pos}(-x) = x$. The powers of the terms look complicated, but they are quite simple; see e.g. Figure 5.1 on page 138 for a graphical depiction, as well as Example 5.17.

Theorem 5.13. *The module $\mathcal{W}_{s,\ell}$ is generated as an $\mathbb{F}[x]$-module by the $\ell + 1$ polynomials $P^{(j)} \in \mathbb{F}[x][y : z]^\ell$ given for $j = 0, \ldots, \ell$ by*

$$P^{(j)} = (g_z y - h_y z)^{a_1(j)} (yz - R_y z^2)^{a_2(j)} (z \tfrac{G}{g_z})^{a_3(j)} y^{\mathrm{pos}(j-s)} z^{\mathrm{pos}(\ell-s-j)}$$

where

$$a_1(j) = \mathrm{pos}(j - (\ell - s)) \qquad a_2(j) = j - \mathrm{pos}(j - (\ell - s)) - \mathrm{pos}(j - s)$$
$$a_3(j) = \mathrm{pos}(s - j)$$

Proof. First, it should be proved that each $P^{(j)}$ is indeed in $\mathbb{F}[x][y : z]^\ell$, i.e. is of total degree ℓ in y and z. By summing all the terms' exponents, counting a_2 twice, and using the identity for $\mathrm{pos}(\cdot)$ given above, one sees this is so.

To show that each $P^{(j)}$ is in $\mathcal{W}_{s,\ell}$, we first show that each of the three terms $yz - R_y z^2$, $z\frac{G}{g_z}$ and $g_z y - h_y z$ interpolate each $(x_i, y_i : z_i)$ once. In the first case, then

$$y_i z_i - R_y(x_i) z_i^2 = (y_i - y_i z_i) z_i$$

and since $z_i \in \{0, 1\}$, the above is always 0. For $z\frac{G}{g_z}$ then either $z_i = 0$ in which case we obviously get 0, or $z_i = 1$ but then $g_z(x_i) \neq 0$ and $G(x_i) = 0$, so we also get 0. In the third case, we have

$$g_z(x_i) y_i - h_y(x_i) z_i = (q_1(x_i) G(x_i) + q_2(x_i) R_z(x_i)) y_i - q_2(x_i) y_i z_i = 0$$

Now, for each $P^{(j)}$ to interpolate the points with multiplicity at least s, we need then only to verify that $a_1(j) + a_2(j) + a_3(j) \geq s$; it is quickly seen that equality holds.

We are then left to show that any $Q \in \mathcal{W}_{s,\ell}$ can be expressed as an $\mathbb{F}[x]$-combination of the $P^{(j)}$. Not surprisingly when looking at Figure 5.1, the argument is divided

into two cases: $\ell - s \leq s$ and $\ell - s > s$. We will only show the latter case, and the former is simpler but follows similarly. So assume $\ell - s > s$. Observe that $\deg_y P^{(j)} = a_1(j) + a_2(j) + \mathrm{pos}(s-j) = j$ by using the aforementioned identity for $\mathrm{pos}(\cdot)$. The proof now basically follows the multivariate division algorithm on Q under lexicographical ordering $y > z > x$; i.e. dividing with the aim of lowering the y-degree. In the following, for some $P \in \mathbb{F}[x][y:z]^\ell$, let us say "leading coefficient" when we mean $P_{[\deg_y P]}(x)$.

First observe that the leading coefficient of $P^{(\ell)}$ is $g_z(x)^s$. By Lemma 5.12, we can perform polynomial division of Q by $P^{(\ell)}$ and get a remainder $Q^{(\ell-1)}(x, y:z)$ of y-degree at most $\ell - 1$. As $P^{(\ell)} \in \mathcal{W}_{s,\ell}$ so is $Q^{(\ell-1)} \in \mathcal{W}_{s,\ell}$. We can continue as such with $P^{(j)}$ for $j = \ell-1, \ell-2, \ldots, \ell-s+1$, as each of these has leading coefficient $g_z(x)^{j-(\ell-s)}$ and Lemma 5.12 promises that the remainders $Q^{(j)}$ will also keep having leading coefficient divisible by exactly this. We thus end with a remainder $Q^{(\ell-s)}$ with y-degree at most $\ell - s$ and in $\mathcal{W}_{s,\ell}$.

As $\ell - s > s$ then for $j = \ell - s, \ldots, s$ we have $P^{(j)}(x, y:z) = (yz - R_y z^2)^s y^{j-s} z^{\ell-s-j}$. They all have leading coefficient 1, so we can reduce $Q^{(\ell-s)}$ with $P^{(\ell-s)}$, reduce the remainder of that with $P^{(\ell-s-1)}$ and so forth, until we arrive at a remainder $Q^{(s-1)}$ with y-degree at most $s-1$.

From $j = s-1$ and downwards, the leading coefficient of $P^{(j)}$ is $(\frac{G}{g_z})^{s-j}$. But by Lemma 5.12, $Q^{(s-1)}$ also has leading coefficient $\frac{G}{g_z}$, so we can reduce it by $P^{(s-1)}$; the remainder $Q^{(s-2)}$ will have y-degree $s-2$, hence leading coefficient $(\frac{G}{g_z})^2$ by Lemma 5.12, and so can be reduced by $P^{(s-2)}$, and so on. At last, we find a remainder $Q^{(0)}$ whose leading coefficient must be divisible by $(\frac{G}{g_z})^s z^\ell = P^{(0)}$. □

As in Section 3.2, we can represent the basis of Theorem 5.13 by an $(\ell+1) \times (\ell+1)$ matrix over $\mathbb{F}[x]$, whose jth row correspond to $P^{(j)}$ and whose tth column correspond to the coefficients to $y^t z^{\ell-t}$ in these polynomials. That is:

$$D_{s,\ell} \in \mathbb{F}[x]^{(\ell+1)\times(\ell+1)} \text{ where } [\, D_{s,\ell} \,]_{j,t} = P^{(j)}_{[t]}(x) \tag{5.1}$$

We will not write out $D_{s,\ell}$ explicitly, as in (3.3) on page 61, since the expressions become rather long and provides little added intuition (also, we would have to discern the two cases $\ell - s > s$ and $\ell - s \leq s$), but see Example 5.17 on page 139.

Note that for any $Q = \sum_{t=0}^{\ell} Q_t(x) y^t z^{\ell-t}$ then $\deg_{1,\theta_1,\theta_2} Q = \deg \Phi_{1,\tilde{\boldsymbol{w}}}\big((Q_0, \ldots, Q_\ell)\big)$, where $\tilde{\boldsymbol{w}} = \big(\ell\theta_2,\ \theta_1 + (\ell-1)\theta_2,\ \ldots,\ (\ell-1)\theta_1 + \theta_2,\ \ell\theta_1\big)$ and $\Phi_{\nu,\boldsymbol{w}}$ are from Definition 2.11 on page 16. We are thus looking for a vector in the module spanned by the rows of $D_{s,\ell}$ which is minimal under the module monomial ordering $\preceq_{1,\tilde{\boldsymbol{w}}}$. However, since θ_1, θ_2 are allowed to be non-integer, so are the entries of $\tilde{\boldsymbol{w}}$, but this is not supported by the module minimisation methods of Chapter 2. Therefore, notice that if $\deg \Phi_{1,\tilde{\boldsymbol{w}}}\big((Q_0, \ldots, Q_\ell)\big) < s\tau$ then also $\deg \Phi_{1,\boldsymbol{w}}\big((Q_0, \ldots, Q_\ell)\big) < s\tau$, where $\boldsymbol{w} = \big(\lfloor w_1 \rfloor,\ \ldots, \lfloor w_{\ell+1} \rfloor\big)$, which means we can use $\boldsymbol{w}$ as weight vector instead.

By Corollary 2.15 on page 18, if $B_{s,\ell}$ is a basis of $\mathcal{W}_{s,\ell}$ in $\Phi_{1,\boldsymbol{w}}(B_{s,\ell})$-weighted weak Popov form, then the row of $B_{s,\ell}$ which orders minimal according to $\preceq_{1,\boldsymbol{w}}$ must

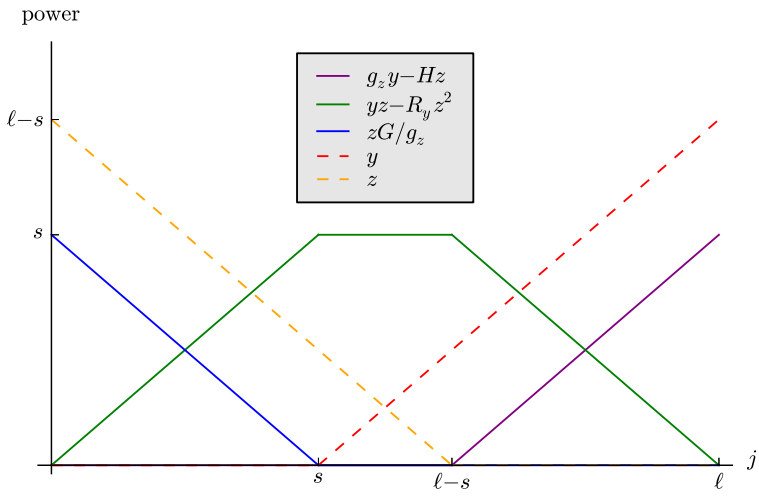

Figure 5.1: Depiction of the powers for the various terms in the elements of the explicit basis $P^{(0)}, \dots, P^{(\ell)}$, assuming that $s < \ell - s$. One can easily verify that the total degree of y and z must be ℓ by inspecting these graphs, remembering that each power of $(yz - R_z z^2)$ contributes two in this degree. It is also easy to see that the total power of $(g_z y - h_y z)$, $(yz - R_z z^2)$ and $z\frac{G}{g_z}$ is s.

exactly be a vector with minimal $(1, \boldsymbol{w})$-weighted degree. Thus, we can solve the problem by applying any of the algorithms in Chapter 2 which can bring $\Phi_{1,\boldsymbol{w}}(D_{s,\ell})$ to weak Popov form.

As usual, we will therefore need some key properties of $D_{s,\ell}$ in order to estimate the complexities of module minimisation. We will not do this as precisely as we did in Lemma 3.25 on page 61, since this case is more involved, and since it does not seem bring us an asymptotic benefit.

Lemma 5.14. *Let* $\boldsymbol{w} = \big(\lfloor \ell\theta_2 \rfloor, \lfloor \theta_1 + (\ell-1)\theta_2 \rfloor, \dots, \lfloor (\ell-1)\theta_1 + \theta_2 \rfloor, \lfloor \ell\theta_1 \rfloor\big)$*, and let* $\hat{\theta} = \max\{\theta_1, \theta_2\}$*. Then*

$$\begin{aligned} \operatorname{maxdeg}\left(\Phi_{1,\boldsymbol{w}}(D_{s,\ell})\right) &\leq sn + \ell\hat{\theta} \\ \Delta(\Phi_{1,\boldsymbol{w}}(D_{s,\ell})) &\in O(s\ell n + \ell^2\hat{\theta}) \end{aligned}$$

Proof. Each entry of $D_{s,\ell}$ is some element in $\mathbb{F}$ times a multiple of g_z, R_y, G and h_y to a combined power of s; since each of the four polynomials have degree at most n, the max-degree follows, since each column is weighted by at most $x^{\ell\hat{\theta}}$ by $\Phi_{1,\boldsymbol{w}}$. For the orthogonality defect, note that $\deg(\Phi_{1,\boldsymbol{w}}(D_{s,\ell})) \leq (\ell+1)\operatorname{maxdeg}\left(\Phi_{1,\boldsymbol{w}}(D_{s,\ell})\right) \leq (\ell+1)sn + \ell(\ell+1)\hat{\theta}$. This is already within the estimate that we have given, and deducting the degree of the determinant to get the orthogonality defect can only decrease this number. □

Lemma 5.15. *Computing* $D_{s,\ell}$ *can be done in* $O\big(\ell \mathrm{P}(s^2 n) + \mathrm{P}(n)\log n\big)$.

Complexity for finding Q for rational interpolation Theorem 5.4 by minimising $D_{s,\ell}$

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^3 s^2 n^2$	$\ell^3 s^2 n^2$
Alekhnovich	$\mathrm{M}(\ell)\mathrm{P}(\ell sn)\log(\ell sn) + \ell^2\mathrm{P}(sn)$	$\ell^4 sn \log(n)^{2+o(1)}$
GJV	$\mathrm{M}(\ell)\mathrm{P}(sn)\log(\ell sn)^{O(1)}$	$\ell^3 sn \log(n)^{O(1)}$

Table 5.1: For relaxation, we have used the same rules as in Table 3.1 on page 63.

Proof. Computing R_y, R_z and G by Lagrangian interpolation can be done in complexity $O(\mathrm{P}(n)\log n)$, see e.g. [vzGG03, p. 297]. g_z and h_y can be computed using the D&C variant of the Euclidean algorithm, which is a special case of the Alekhnovich algorithm in $O(\mathrm{P}(n)\log n)$, see Table 2.4 on page 42. By [vzGG03, Corollary 8.28], multiplying together two polynomials in $\mathbb{F}[x,y]$ of x-degree at most N and y-degree at most ℓ, costs $\mathrm{P}(\ell N)$; clearly the price is the same for multiplying together polynomials in $\mathbb{F}[x][y:z]^\ell$. Thus, calculating the first s powers of each of $(g_z y - h_y z)$, $(yz - R_z z^2)$ and $z\frac{G}{g_z}$ costs $O(s\mathrm{P}(s^2 n))$. After this, all the $P^{(j)}$ can be calculated in $O(\ell\mathrm{P}(s^2 n))$. With proper memory management, the at most s non-zero $y^j z^{\ell-j}$ coefficients of each can be extracted out to $D_{s,\ell}$ in total complexity $O(\ell s)$ using pointers. □

Remark. Note that the above is slightly slower than computing $A_{s,\ell}$, see Lemma 3.26 on page 62. I have been unable to find a construction which is as fast, in particular since some of the entries of $D_{s,\ell}$ are *sums* of terms of the form $g_z^{\mu_1} h_y^{\mu_2} R_y^{\mu_3} (\frac{G}{g_z})^{\mu_4}$. ◆

Theorem 5.16. *For a given instance of the rational interpolation problem with $\tau^2 > n(\theta_1+\theta_2)$, and using the parameters of Proposition 5.7 on page 131, we can find a satisfactory Q in the complexity as given in Table 5.1, for various choices of module minimisation algorithm.*

Proof. By the assumption $\tau^2 > n(\theta_1+\theta_2)$, it follows that we *can* employ the parameter choices of Proposition 5.7. From these it also follows that $\ell \in O(s\frac{\tau}{\theta}+1)$ with $\theta = \theta_1+\theta_2$, and so $O(\ell\theta) \subset O(s\tau) \subset O(sn)$. Therefore by Lemma 5.14, $\operatorname{maxdeg}(D_{s,\ell}) \in O(sn)$ and $\Delta(D_{s,\ell}) \in O(\ell sn)$, and the result follows from using Table 2.3 on page 42. Note that by Lemma 5.15, the complexity of constructing $D_{s,\ell}$ is overshadowed by the complexity of minimising it, no matter which minimisation algorithm we choose. □

Remark. Note that in the usual case where both θ_1 and θ_2 are positive, all entries of $\boldsymbol{w}$ are positive and so we can normalise the ordering into $\preceq_{1,\boldsymbol{w}_\circ}$ where $\boldsymbol{w}_\circ = (w_1-\delta, w_2-\delta, \ldots, w_{\ell+1}-\delta)$ where δ is the minimal element of $\boldsymbol{w}$; this is a completely equivalent ordering. Clearly this won't change the orthogonality defect, but it will decrease the max degree, thus slightly improving the running time (though not asymptotic). ◆

Example 5.17. *Consider a rational interpolation problem with parameters*

$$n = 250 \qquad \tau = 105 \qquad \theta_1 = 15 \qquad \theta_2 = 14$$

We have $\theta = 29$. *Proposition 5.7 on page 131 now tells us that* $(s, \ell) = (2, 4)$ *can be used for the rational interpolation problem.*

A satisfactory Q *should have* $(1, 15, 14)$*-weighted degree at most* 210. *Furthermore, its coefficients must reside in the row space of the matrix* $D_{2,4}$. *Assuming that* $z_i = 0$ *for exactly 5 values of* i, *and assuming that* $\deg R_y = \deg h_y = n - 1$, *which is the generic case, then* $D_{2,4}$ *has the form*

$$\begin{pmatrix} (\frac{G}{g_z})^2 & 0 & 0 & 0 & 0 \\ -\frac{G}{g_z} R_y & \frac{G}{g_z} & 0 & 0 & 0 \\ R_y^2 & -2\,R_y & 1 & 0 & 0 \\ 0 & h_y R_y & -R_y g_z - h_y & g_z & 0 \\ 0 & 0 & h_y^2 & -2\,h_y g_z & g_z^2 \end{pmatrix} \trianglelefteq \begin{pmatrix} 490 & \bot & \bot & \bot & \bot \\ 494 & 245 & \bot & \bot & \bot \\ 498 & 249 & 0 & \bot & \bot \\ \bot & 498 & 254 & 5 & \bot \\ \bot & \bot & 498 & 254 & 10 \end{pmatrix}$$

We set $\boldsymbol{w} = (56, 57, 58, 59, 60)$, *and are then to minimise the* $\Phi_{1,\boldsymbol{w}}$ *image of* $D_{s,\ell}$. *We have*

$$\begin{aligned} \operatorname{maxdeg}(\Phi_{1,\boldsymbol{w}}(D_{2,4})) &= 556 \;\; \leq 560 \;\; = sn + \ell\hat{\theta} \\ \deg(\Phi_{1,\boldsymbol{w}}(D_{2,4})) &= 2761 \leq 2800 = (\ell+1)sn + \ell(\ell+1)\hat{\theta} \\ \Delta(\Phi_{1,\boldsymbol{w}}(D_{2,4})) &= 1721 \end{aligned}$$

♠

5.2 Decoding GRS codes

Consider once again some $[n,\ k,\ n-k+1]$ GRS code, as well as some sent codeword $\boldsymbol{c} = \operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f)$ with $\deg f < k$ and some received word $\boldsymbol{r}$. Recall the definitions of G and R from Definition 3.22 on page 60. Choosing the simplest $\ell = 1$ in Corollary 4.3 on page 96, we have the Gao key equation:

$$\begin{aligned} \Lambda R &\equiv \Lambda f \mod G \\ \deg \Lambda + k &> \deg(\Lambda f) \end{aligned}$$

As we have seen so many times before, we can solve this key equation using the methods of Chapter 2: we would be minimising the $\Phi_{1,(k,0)}$-image of the matrix

$$M_{\text{Wu}} = \begin{pmatrix} 1 & R \\ 0 & G \end{pmatrix} \tag{5.2}$$

giving us a matrix $[\frac{\boldsymbol{g}_1}{\boldsymbol{g}_2}] \in \mathbb{F}[x]^{2\times 2}$ in $\Phi_{1,(k,0)}$-weighted weak Popov form; thus $\boldsymbol{g}_1, \boldsymbol{g}_2$ is a Gröbner basis of the row space of M_{Wu} with respect to $\preceq_{1,(k,0)}$. Assume in the following that $\boldsymbol{g}_1, \boldsymbol{g}_2$ has been numbered such that $\operatorname{LP}_{\preceq_{1,(k,0)}}(\boldsymbol{g}_i) = i$ for $i = 1, 2$.

As we know from Proposition 4.4 on page 97, if $|\mathcal{E}| = \deg \Lambda \leq \frac{n-k}{2}$ then $\boldsymbol{g}_1 = \gamma(\Lambda, \Lambda f)$ for some $\gamma \in \mathbb{F}$. Whenever more errors occur, $(\Lambda, \Lambda f)$ will be an $\mathbb{F}[x]$-linear combination of $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$. Wu's observation is how to utilise the knowledge that Λ splits and has roots in $\{\alpha_1, \ldots, \alpha_n\}$ to find this combination by rational interpolation faster than brute force search:

Proposition 5.18. *If $|\mathcal{E}| > \frac{n-k}{2}$ then let $p_1, p_2 \in \mathbb{F}[x]$ be such that $(\Lambda, \Lambda f) = p_1\boldsymbol{g}_1 + p_2\boldsymbol{g}_2$. Then*

$$\deg p_1 = |\mathcal{E}| - \deg g_{1,1} \qquad \deg p_2 < |\mathcal{E}| - \deg g_{2,2} + k$$

Furthermore, let $(y_i : z_i) = (g_{2,1}(\alpha_i) : -g_{1,1}(\alpha_i))$ for $i = 1, \ldots, n$. Then for exactly $|\mathcal{E}|$ out of the n choices of i, we have

$$z_i p_1(\alpha_i) - y_i p_2(\alpha_i) = 0 \tag{5.3}$$

Proof. From Proposition 2.14 on page 17, we immediately get the degree bounds, since $\deg \Lambda = |\mathcal{E}|$. Now, since $\Lambda = p_1 g_{1,1} + p_2 g_{2,1}$, then for $i \in \mathcal{E}$ we have

$$0 = \Lambda(\alpha_i) = p_1(\alpha_i) g_{1,1}(\alpha_i) + p_2(\alpha_i) g_{2,1}(\alpha_i)$$

Therefore, p_1, p_2 must satisfy (5.3) for these i; whenever $i \notin \mathcal{E}$, then $\Lambda(\alpha_i) \neq 0$, so the identity is indeed satisfied exactly $|\mathcal{E}|$ times. Note that the $(y_i : z_i)$ are indeed projective points: we can't have $g_{1,1}(\alpha_i) = 0$ and $g_{2,1}(\alpha_i) = 0$ simultaneously, for $g_{1,1}$ and $g_{2,1}$ must be coprime since there exists a linear combination of $\boldsymbol{g}_1, \boldsymbol{g}_2$ giving $(1, R)$. □

Remark. Since also Λf evaluates to 0 at all the error positions, we could instead have used the evaluations of $g_{2,1}$ and $g_{2,2}$ to define $(y_i : z_i)$. This yields exactly the same: observe that by the structure of M_{Wu}, then $g_{1,2} = g_{1,1}R + q_1 G$ for some $q_1 \in \mathbb{F}[x]$, and similarly $g_{2,2} = g_{2,1}R + q_2 G$. Therefore, for each α_i we have

$$\big(g_{1,2}(\alpha_i) : g_{2,2}(\alpha_i)\big) = \big(g_{1,1}(\alpha_i) r_i' : g_{2,1}(\alpha_i) r_i'\big) \sim \big(g_{1,1}(\alpha_i) : g_{2,1}(\alpha_i)\big)$$

where the $\sim$ is the equivalence relation on points in $\mathbb{P}^1(\mathbb{F})$. ◆

Since the points $(y_i : z_i)$ can be calculated by the receiver once he has computed $\boldsymbol{g}_1, \boldsymbol{g}_2$, clearly Proposition 5.18 then specifies a rational interpolation problem—assuming that the value of $|\mathcal{E}|$ is known. We should then choose θ_1 and θ_2 to be the degree upper bounds on p_1 and p_2. Momentarily we will show using Lemma 5.8 on page 132 that a Q we construct assuming, say, $|\mathcal{E}| = \tau$, will also work when $|\mathcal{E}| < \tau$. So for which values of τ can we find a valid Q? By Proposition 5.7, at least whenever $\tau^2 > n(\theta_1 + \theta_2)$. But assuming $|\mathcal{E}| = \tau$ then we would set

$$\theta = \theta_1 + \theta_2 = 2\tau - (\deg g_{1,1} + \deg g_{2,2}) + k - 1 = 2\tau - d$$

since $\deg g_{1,1} + \deg g_{2,2} = \deg \det M_{\text{Wu}} = n$. So we can construct Q whenever $\tau^2 > n(2\tau - d)$, that is $\tau < n - \sqrt{n(n-d)}$, i.e. exactly the Johnson bound that we saw in Proposition 3.8 on page 52!

The various parts can now be collected into Algorithm 6 on page 143. By "valid error-locator", we mean a polynomial which splits and roots only of multiplicity 1 and all from $\{\alpha_1, \ldots, \alpha_n\}$. Note that we do not require it to be monic. We then have:

Theorem 5.19. *Algorithm 6 is correct.*

Proof. By the discussion in the beginning of this section, if $|\mathcal{E}| \leq \frac{n-k}{2}$, then $g_{1,1} = \Lambda$ in Line 3. However, the algorithm needs to return *all* codewords of distance at most τ, whence we only break early in Line 3 if we are sure there can be no other codewords within this radius: since all other codewords must be at least d from $\boldsymbol{c}$, i.e. at least $d - |\mathcal{E}|$ from $\boldsymbol{r}$, we can safely break early whenever $\tau < d - \deg g_{1,1}$ and correcting using $g_{1,1}$ yields a codeword.

If we do not break in Line 3, we go on to construct a Q. Since $\tau < n - \sqrt{n(n-d)}$ then $\tau^2 > n\theta$ (see above), so we can use the parameter choices of Proposition 5.7 on page 131 and a Q can indeed be constructed. Note that $\tau^2 > n\theta$ implies $\theta < \tau$ so Proposition 5.7 promises parameters satisfying $s \leq \ell$. By Proposition 5.18, if $|\mathcal{E}| = \tau$, it must be the case that p_1, p_2 are among the $p_1^\star, p_1^\star$. If $|\mathcal{E}| < \tau$, then by Lemma 5.8 on page 132 this is also the case as long as

$$\min\{\theta_1 - \deg p_1,\ \theta_2 - \deg p_2\} \geq \tfrac{s}{\ell}(\tau - |\mathcal{E}|)$$

But $\theta_1 - \deg p_1 = \tau - \deg g_{1,1} - (|\mathcal{E}| - \deg g_{1,1}) = \tau - |\mathcal{E}|$, and similarly $\theta_2 - \deg p_2 = \tau - |\mathcal{E}|$. Therefore, the above is true since $s \leq \ell$. Note that in Line 7, we need not check the distance of the calculated words to $\boldsymbol{r}$, since their distance will be exactly the degree of the respective $\Lambda^\star$; and this can in turn not be greater than $\deg p_{1,1}^\star + \deg g_{1,1} \leq \tau$. □

To round off, let us sum up the asymptotic complexity of the method. Not surprisingly, the total cost is completely dominated by module minimisation.

Corollary 5.20. *Using the root-finding algorithm of Proposition 5.9, and for various choices of module minimisation algorithms for construction of Q, then Algorithm 6 has the complexity as given in Table 5.2, assuming that $q \in O(sn)$ where q is the size of the field.*

Proof. The only step of Algorithm 6 which we have not already addressed is the checks for "valid error-locator", finding the error positions, i.e. roots of the error locators, as well as performing error correction. For a given polynomial $\lambda(x)$, the trivial method of determining whether it is a valid error-locator is to evaluate it at all α_i and see if this yields exactly $\deg \lambda$ roots. All these n evaluations can be done using Fast Fourier transform in $O(q \log q)$ where q is the size of the field. Since we will have one candidate to check in Line 3, and at most ℓ to check in Line 5, the total cost will be in $O(\ell q \log q)$. □

[2] In [BHNW13, p.3276], this erroneously read "at most".
[3] In [BHNW13, p.3276], these degree bounds erroneously read "less than".

Algorithm 6 Wu list decoding GRS codes

Input: The received word $\boldsymbol{r} = (r_1, \ldots, r_n)$. Parameters s, ℓ, τ such that $\tau < n - \sqrt{n(n-d)}$ and $E_{\text{Wu}}^{[n,2\tau-d]}(s,\ell,\tau) > 0$ as well as $s \leq \ell$ for the given code, e.g. by using Proposition 5.7 on page 131.

Output: A list of all codewords in $\mathcal{C}$ within radius τ of $\boldsymbol{r}$ or Fail if there are no such words.

1 Calculate G and R according to Definition 3.22 on page 60.
2 Compute $\boldsymbol{g}_1, \boldsymbol{g}_2$ by module minimising M_{Wu} of (5.2), with $\text{LP}_{\preceq_{1,(k,0)}}(\boldsymbol{g}_1) = 1$.
3 If $g_{1,1}$ is a valid error-locator of degree less than[2] $d - \tau$, use it to correct $\boldsymbol{r}$, and if this yields a word in $\mathcal{C}$, return this one word.
4 Otherwise, set $\theta_1 = \tau - \deg g_{1,1}$ and $\theta_2 = \tau - \deg g_{2,2} + k - 1$. Construct a $Q(x, y : z)$ satisfying the requirements of Theorem 5.4 on page 130 using the points $\{(\alpha_i, g_{1,1}(\alpha_i) : -g_{2,1}(\alpha_i))\}_{i=1}^n$ and using the parameters s, ℓ, τ.
5 Find all $p_1^\star, p_2^\star$ such that[3] $\deg p_1^\star \leq \theta_1$ and $\deg p_2 \leq \theta_2$ and $Q(x, p_1^\star, p_2^\star) = 0$. Return Fail if no such factors exist.
6 For each such factor, construct $\Lambda^\star(x) = p_1^\star(x) g_{1,1}(x) + p_2^\star(x) g_{2,1}(x)$. If it is a valid error-locator, use it for correcting $\boldsymbol{r}$. Return Fail if none of the factors yield error-locators.
7 Return those of the corrected words that are in $\mathcal{C}$. Return Fail if there are no such words.

Remark. If one is not interested in a list decoder per se, but rather in a maximum-likelihood list decoder (such as is considered in Section 3.3), then one can safely break in Line 3 if $g_{1,1}$ is an error locator and has degree up to $d/2$; for clearly there can be no other words within half the minimum distance. ◆

Remark. Consider a GRS code with odd minimum distance d. Choosing $\tau = d/2$, i.e. exactly one more error than minimum-distance decoding, yields the "+1 decoder"; this is an important special case since it can decode an additional error in basically the same complexity as minimum-distance decoding. It was pointed out by Wu in [Wu08][4].

In this case $\theta = 0$. Consider now that τ errors occurred. We know that $(\Lambda, \Lambda f) = p_1 \boldsymbol{g}_1 + p_2 \boldsymbol{g}_2$, with $p_1, p_2 \in \mathbb{F}[x]$ and neither non-zero, so $\theta_1, \theta_2 \geq 0$. But since $\theta_1 + \theta_2 = \theta = 0$ then also $\theta_1 = \theta_2 = 0$, i.e. $\deg g_{1,1} = \tau$ and $\deg g_{2,2} = \tau + k - 1$. We are therefore actually just looking for *constants* which "evaluate" to exactly τ of the $(g_{1,1}(\alpha_i) : -g_{2,1}(\alpha_i))$.

Clearly, there is no need for rational interpolation here: we calculate all the points $(g_{1,1}(\alpha_i) : -g_{2,1}(\alpha_i))$, and bundle them into groups if they are equivalent, i.e. two such points $a, b \in \mathbb{P}_2(\mathbb{F})$ are put in the same group if $a \sim b$. Each such group gives rise to a possible error locator by picking one of the elements and treating this as

[4] It is possible that ideas similar to these were already used in Berlekamp's +1 decoder for soft-decision decoding [Ber96], though I have not studied this paper well enough to be sure.

Complexity for Wu list decoding GRS codes

Algorithm	Complexity	Relaxed
Mulders–Storjohann	$\ell^3 s^2 n^2$	$\ell^3 s^2 n^2$
Alekhnovich	$\mathrm{M}(\ell)\mathrm{P}(\ell sn)\log(\ell sn) + \ell^2\mathrm{P}(sn)$	$\ell^4 sn\log(n)^{2+o(1)}$
GJV	$\mathrm{M}(\ell)\mathrm{P}(sn)\log(\ell sn)^{O(1)}$	$\ell^3 sn\log(n)^{O(1)}$

Table 5.2: For relaxation, we have used the same rules as in Table 3.1 on page 63.

$(p_1^\star : p_2^\star)$; we then run Algorithm 6 from Line 5.

Before doing this, we can easily prune the list of candidates in order to minimise wasted effort: if $g_{1,1}$ was not an error-locator in Line 3, then we know that $|\mathcal{E}| > \frac{n-k}{2}$ so $|\mathcal{E}| = \tau$; thus we should consider only groups of points containing exactly τ elements. Even if $g_{1,1}$ was an error locator, we know that all other valid error locators must have $\deg\Lambda \geq \deg g_{1,1}$ as well as $\deg\Lambda \geq d - \deg g_{1,1}$; thus we can remove groups not satisfying this.

The complexity of this is just the complexity of Algorithm 6 up to and including Line 4, followed by the judging of around $\lfloor \frac{n}{\tau} \rfloor$ error locators in the worst case. ♦

Example 5.21. *Consider again the [250, 70, 181] code of Example 3.5 on page 50, and we want to decode up to $\tau = 105$ errors. Assume that more than half the minimum distance errors occurred, i.e. $|\mathcal{E}| > 90$. After $\Phi_{1,(k,0)}$-weighted module minimisation of M_{Wu}, the degrees of $\boldsymbol{g}_1, \boldsymbol{g}_2$ would depend on the instance, but in the usual case we obtain:*

$$\begin{pmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{pmatrix} \trianglelefteq \begin{pmatrix} 90 & 159 \\ 90 & 160 \end{pmatrix}$$

Assuming this, we would calculate $\theta_1 = 15$ and $\theta_2 = 14$. We confirm that $\theta = \theta_1 + \theta_2 = 29 = 2\tau - d$. The resulting rational interpolation problem has exactly the parameters we explored in Example 5.17; in particular $(s, \ell) = (2, 4)$. It is therefore interesting to compare the size of the module minimisation problem with the one we would solve for Guruswami–Sudan, in either Example 3.28 on page 62 or Example 3.39 on page 74, for this case where the parameters are all the same. We can see that the metrics of the matrices of Example 3.28 and this example are very comparable, while that of Example 3.39 on page 74 is somewhat larger. ♠

5.2.1 Using the syndrome key equation

In [Wu08], Wu originally proposed his algorithm using the syndrome key equation. We will briefly show how this is completely analogous to using the Gao key equation above. Choosing $\ell = 1$ in Corollary 4.8 on page 102, we have the syndrome key

equation:

$$\begin{aligned} \overline{\Lambda} S &\equiv \overline{\Omega} \mod x^{n-k} \\ \deg \overline{\Lambda} &> \deg \overline{\Omega} \end{aligned}$$

where $\overline{\Omega} = \overline{\Omega}^{(1)}$ and $S = (S_\bullet^{(1)} \bmod x^{n-k})$. Again, we would solve this key equation by minimising the matrix

$$M_{\text{Wu}}^S = \begin{pmatrix} 1 & S \\ 0 & x^{n-k} \end{pmatrix}$$

To be clear, we would minimise its $\Phi_{1,\mathbf{0}}$-image, but $\Phi_{1,\mathbf{0}}$ is the identity function. Let $\left[\frac{\tilde{\boldsymbol{g}}_1}{\tilde{\boldsymbol{g}}_2}\right]$ be this basis, and such that $\text{LP}(\tilde{\boldsymbol{g}}_1) = 1$. In the row space of this matrix must be $(\overline{\Lambda}, \overline{\Omega})$. If $|\mathcal{E}| \leq \frac{n-k}{2}$, we even have $\gamma(\overline{\Lambda}, \overline{\Omega}) = \tilde{\boldsymbol{g}}_1$ for some $\gamma \in \mathbb{F}$.[5]

Completely as before, if more errors have occurred we then know that $(\overline{\Lambda}, \overline{\Omega}) = p_1\tilde{\boldsymbol{g}}_1 + p_2\tilde{\boldsymbol{g}}_2$ for some $p_1, p_2 \in \mathbb{F}[x]$. We also get analogous degree constraints on p_1 and p_2, and using that $\overline{\Lambda}$ must be zero on the inverse of all the error positions, we can thus set up a rational interpolation problem for which p_1, p_2 is a solution. All parameters θ, τ, s, ℓ for the rational interpolation problem would be as before, and therefore also the complexity of the rational interpolation would be unchanged.

5.2.2 Relation to Guruswami–Sudan in choice of parameters

The parameter choices for the rational interpolation problem in the Wu list decoder were given in Proposition 5.7 on page 131 and found by remarking that $E_{\text{Wu}}^{[n,\theta]}(s, \ell, \tau) = E_{\text{GS}}^{[n,\theta+1]}(s, \ell, n-\tau)$, and then using the parameters of Proposition 3.11 on page 53.

In the case of GRS decoding we have yet another, more direct, duality. Here we have $\theta = 2\tau - d$, and by insertion, one sees that

$$E_{\text{Wu}}^{[n,2\tau-d]}(s, \ell, \tau) = E_{\text{GS}}^{[n,k]}(\ell - s, \ell, \tau) \tag{5.4}$$

Written out in words, and elaborated upon, one can then very directly compare the list sizes and multiplicities which are possible in the Guruswami–Sudan with those in Wu's algorithm:

Proposition 5.22. *For a given code, and some decoding radius $\tau \geq \frac{n-k+1}{2}$, then s and ℓ are valid choices for the parameters for the Wu list decoder if and only if ℓ and $s_{\text{GS}} = \ell - s$ are valid choices for the Guruswami–Sudan.*

Furthermore, for a given ℓ, let s be minimal such that $E_{\text{Wu}}^{[n,2\tau-d]}(s, \ell, \tau) > 0$, and s_{GS} minimal such that $E_{\text{GS}}^{[n,k]}(s_{\text{GS}}, \ell, \tau) > 0$. If $\tau < n/2$ then $s \leq s_{\text{GS}}$, otherwise, $s \geq s_{\text{GS}}$.

[5] This easily follows from classical views on the key equation, e.g. [Dor87, SKHN75]. However, in our exposition it follows from Proposition 4.10 on page 103 since we proved in Proposition 4.4 on page 97 that Power Gao always succeeds when $|\mathcal{E}| \leq \frac{n-k}{2}$; also when $\ell = 1$.

Proof. Clearly, the first claim follows directly from (5.4). For the second claim, consider $E_{\text{Wu}}^{[n,2\tau-d]}(s,\ell,\tau) = (\ell+1)s\tau - \binom{\ell+1}{2}(2\tau-d) - \binom{s+1}{2}n$. For given τ and ℓ then requiring this to be positive gives a second degree equation in s; rearranging and solving we get that s must be chosen from the interval centred at ℓT and with length $2\ell\tilde{D}^{1/2}$, where $T = \frac{\tau}{n} + \frac{\tau - n/2}{n\ell}$ and the precise expression of $\tilde{D}$ is not important for us. Due to (5.4), then the corresponding interval for valid s_{GS} must be centred at $\ell(1-T)$ and also has length $2\ell\tilde{D}^{1/2}$. In addition to residing in these respective intervals, we only require of s and s_{GS} that they are positive integers less than or equal to ℓ. Therefore, whenever $\tau < n/2$ we have $T < \frac{1}{2}$, so the smallest possible choice of s in the former interval must be at most the smallest possible in the latter interval; oppositely for the case $\tau \geq n/2$. □

Remark. The above fits almost comically well with a general intuition: in the Guruswami–Sudan, we are performing an interpolation on the *true positions* since we are guessing $f(x)$. In Wu's list decoder, on the other hand, we are interpolating the *error positions*, since we are essentially guessing $\Lambda(x)$. The above proposition says that as long as there are more true positions than error positions, then it is "cheaper" to locate the latter and vice versa! And by "cheaper", we just mean that the multiplicity is lower.

In general, medium to high-rate codes are used more often than very low-rate ones, which means that we are usually correcting far fewer errors than $n/2$. What is also revealed from the above proof is that the ratio s_{GS}/s increase with the rate, so the benefit, when considering only this measure, of using the Wu list decoder in place of the Guruswami–Sudan increases for higher rate codes. ♦

Example 5.23. *Consider again the* [2480, 1489, 992] *code of Example 3.16 on page 58. Decoding up to* $\tau = 558$ *errors, we can use* $\ell = 280$, $s = 63$ *and* $s_{\text{GS}} = 217$*; these are what Proposition 3.11 on page 53 and Proposition 5.7 on page 131 evaluate to, and also the minimal possible. These parameters might still be too large for practical use, but it demonstrates that the saving in multiplicity for Wu's list decoder is significant compared to the Guruswami–Sudan, even for medium to low-rate codes.* ♠

Example 5.24. *Consider a quite high-rate* [2047, 1800, 248] *GRS code. Minimum-distance decoding is* 124*, while* $\lceil n - \sqrt{n(n-d)} - 1 \rceil = 128$ *is the list-decoding radius. Decoding up to* $\tau = 127$*, we can use* $\ell = 62$, $s = 3$ *while* $s_{\text{GS}} = 59$*; these are the minimal possible, and are also obtained from Proposition 5.7 as well as the duality; using Proposition 3.11, however, gives* $\ell = 63$ *and* $s_{\text{GS}} = 60$*. Thus for such high-rate codes the saving is quite substantial.* ♠

Remark. The duality of (5.4) could also be used to obtain closed expression for the parameters for Wu list decoding GRS codes from those for Guruswam–Sudan in Proposition 3.11 on page 53. The two are not exactly the same, and as we see above they do not in all cases give exactly the same result. ♦

5.3 Wu list decoding binary Goppa codes

In this section we will show how binary Goppa codes can be list decoded very well using the Wu approach. Goppa codes are a beautiful class of algebraic codes, first described in [Gop70]; especially in the binary case, the codes have very good minimum distance (for not too long codes). Sugiyama et al. [SKHN75] gave a key equation for q-ary Goppa codes and described how to use the extended Euclidean algorithm to solve this; Patterson [Pat75] rewrote this into one "modulo x^N" in order to let it be solvable by the Berlekamp–Massey algorithm, and in the same publication gave an important rewriting of the key equation in the case of binary Goppa codes to take full advantage of the better minimum distance. We will use the same rewriting here. For a wholesome description, along with proofs of some of the basic facts on Goppa codes used below, see e.g. [MS77].

Consider an irreducible polynomial $G(x) \in \mathbb{F}_{2^m}[x]$ as well as n distinct elements of $\mathbb{F}_{2^m}$, $L = (\alpha_1, \ldots, \alpha_n)$. Then the irreducible binary Goppa code $\Gamma(G, L)$ with Goppa polynomial G over L is the set

$$\mathcal{C} = \left\{ (c_1, \ldots, c_n) \in \mathbb{F}_2^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \equiv 0 \mod G(x) \right\}$$

It can be shown that this code has parameters $[n, \geq n - m \deg g, \geq 2 \deg g + 1]$.

As usual, consider a sent word $\boldsymbol{c} \in \mathcal{C}$ and a received $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$, and let $\mathcal{E} = \{i \mid e_i \neq 0\}$. For Goppa codes, a natural definition of a syndrome polynomial is then

$$S(x) = \left(\sum_{i=1}^{n} \frac{r_i}{x - \alpha_i} \mod G(x) \right) = \left(\sum_{i=1}^{n} \frac{e_i}{x - \alpha_i} \mod G(x) \right)$$

It is then quite natural to introduce an error-locator and error-evaluator:

$$\Lambda(x) = \prod_{i \in \mathcal{E}} (x - \alpha_i) \qquad \Omega(x) = \sum_{i \in \mathcal{E}} \prod_{j \in \mathcal{E} \setminus \{i\}} (x - \alpha_j)$$

Notice that we again have $\gcd(\Lambda, \Omega) = 1$.

Proposition 5.25. $\Lambda(x) S(x) \equiv \Omega(x) \mod G(x)$

Proof. By insertion. □

Note that for a binary code, the receiver can decode immediately upon having calculated the error locator, even without the error evaluator since $e_i \neq 0 \implies e_i = 1$.

Now we could proceed exactly as in the case of GRS codes in Section 5.2, and we would arrive at a list decoder correcting up to $J(n, \deg G + 1) = n - \sqrt{n(n - \deg G - 1)}$ errors. However, this radius is much less than $\deg G$ which is promised by the minimum distance of the binary Goppa code, and which can be corrected by Patterson's decoder [Pat75]; see Figure 5.2 on page 148.

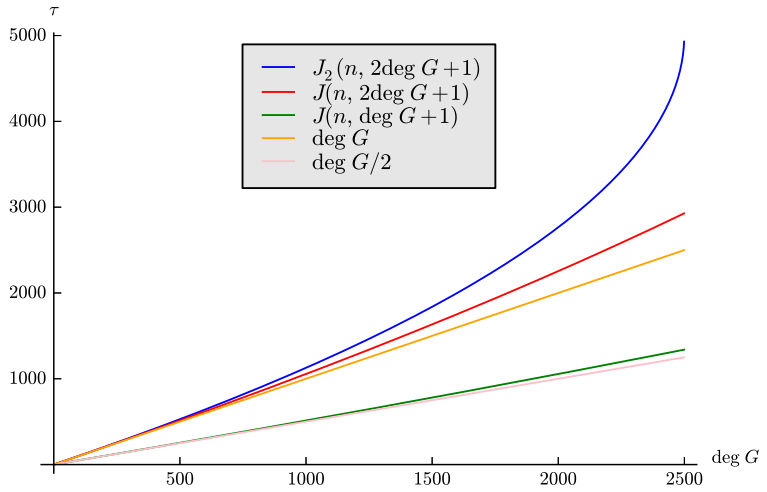

Figure 5.2: Depiction of various decoding radii for the range of binary Goppa codes with $n = 10\,000$ and various $\deg G$. Our decoder decodes to $J_2(n, 2\deg G + 1)$, i.e. the best, while unique decoding with Patterson is up to $\deg G$. $J(n, \deg G + 1)$ is achieved by just using Guruswami–Sudan on the GRS code surrounding the Goppa code, or by applying the Wu method on the initial key equation Proposition 5.25, while $\deg G/2$ is achieved by minimum distance decoding the surrounding GRS code. The last, $J(n, 2\deg G + 1)$ can be achieved by methods discussed in Section 5.3.2 and Section 5.4: by applying the Guruswami–Sudan on the GRS code $\Gamma(G^2, L)$, or by employing Bernstein's (first) decoder. Be wary of this picture: a priori, a binary Goppa code with Goppa polynomial $\deg G$ usually has dimension $n - m\deg G$ where $m = \lceil \log_2 n \rceil = 14$ in this example. Thus for any random G with $\deg G > 714$, the code most likely has dimension 0.

Therefore, we proceed first with Patterson's rewrite of the key equation.

Lemma 5.26. *Assume* $|\mathcal{E}| < 2\deg G + 1$. *If* $S^{-1}(x) \equiv x \mod G(x)$ *and* $0 \in L$, *then* $\Lambda(x) = x$. *Otherwise, let* $a, b \in \mathbb{F}[x]$ *be such that* $\Lambda = a^2 + xb^2$. *Then* a, b *satisfy* $\deg a \leq \lfloor \frac{\mathcal{E}}{2} \rfloor$ *and* $\deg b \leq \lfloor \frac{\mathcal{E}-1}{2} \rfloor$, *as well as*

$$b(x)\tilde{S}(x) \equiv a(x) \mod G(x) \tag{5.5}$$

where $\tilde{S}$ *is the unique polynomial such that* $\tilde{S}^2 \equiv x + S^{-1} \mod G$ *and* $\deg \tilde{S} < \deg G$.

Proof. The a and b are simply formed by collecting even and odd terms of Λ and clearly exist as well as satisfy the degree constraints. Furthermore, since we are working over $\mathbb{F}_2$, note that Ω equals the formal derivative of Λ, which means $\Omega(x) = b^2(x)$. The key equation thus becomes

$$\begin{aligned}(a^2(x) + xb^2(x))S(x) &\equiv b^2(x) \mod G(x) \iff \\ b^2(x)(x + S^{-1}(x)) &\equiv a^2(x) \mod G(x)\end{aligned} \tag{5.6}$$

Note here that calculating the inverse of $S(x)$ modulo $G(x)$ is possible since $\deg S < \deg G$ and $G(x)$ is irreducible.

It might now be that $S^{-1}(x) \equiv x \mod G(x)$ in which case $a^2(x) \equiv 0 \mod G(x)$. As $G(x)$ is irreducible, a must then be a multiple of G, which means that $a = 0$ as $\deg a \leq \mathcal{E}/2 < \deg G$. This implies $\Lambda = xb^2$, but since Λ is squarefree, then this is a legal error locator only when $0 \in L \cap \mathcal{E}$ and $b(x) = 1$.

Having taken care of the case $S^{-1}(x) \equiv x \mod G(x)$, let us now assume that this is not the case and continue. As $G(x)$ is irreducible, $\mathbb{F}_{2^m}[x]/\langle G(x)\rangle$ is a finite field of characteristic 2, so we can compute a square-root; in particular, the described $\tilde{S}(x)$ can be calculated. Inserting $\tilde{S}(x)$ in (5.6), we get

$$\begin{aligned} b^2(x)\tilde{S}^2(x) &\equiv a^2(x) \mod G(x) \iff \\ b(x)\tilde{S}(x) &\equiv a(x) \mod G(x) \end{aligned}$$

finishing the proof □

We are therefore in the case of a new key equation, where the degrees of the unknown polynomials are halved! Note that $\tilde{S}$ is directly computable by the receiver after having computed S. Proposition 5.25 tells us that (a, b) is in the $\mathbb{F}[x]$ rowspan of the matrix

$$M_{\text{Wu}}^{\Gamma} = \begin{pmatrix} 1 & \tilde{S} \\ 0 & G \end{pmatrix} \tag{5.7}$$

For any $\boldsymbol{w} \in \mathbb{N}^2$, we can perform $\Phi_{1,\boldsymbol{w}}$-weighted module minimisation of M_{Wu}^{Γ} and find a Gröbner basis with respect to $\preceq_{1,\boldsymbol{w}}$. Optimal decoding radius turns out to be obtained by choosing $\boldsymbol{w} = \mathbf{0}$. Since it is neither a nor b we know has a lot of zeroes among the α_i, but Λ constructed by them, the road to a rational interpolation problem in this case has an additional turn compared to the case for GRS decoding. The analogue of Proposition 5.18 on page 141 becomes:

Proposition 5.27. *Let a, b be as in Lemma 5.26. Let $\left[\frac{\boldsymbol{g}_1}{\boldsymbol{g}_2}\right]$ be a basis in weak Popov form of the row space of M_{Wu}^{Γ}, having $\text{LP}(\boldsymbol{g}_1) = 1$.*

If $|\mathcal{E}| \leq \deg G$ then $(a, b) = \gamma \boldsymbol{g}_1$ or $(a, b) = \gamma \boldsymbol{g}_2$ for some $\gamma \in \mathbb{F}_{2^m}$.

If instead $\deg G < |\mathcal{E}| < 2 \deg G + 1$ then let $p_1, p_2 \in \mathbb{F}[x]$ be such that $(a, b) = p_1 \boldsymbol{g}_1 + p_2 \boldsymbol{g}_2$. Then p_1 and p_2 are coprime and:

$$\begin{aligned} &\deg p_1 = \tfrac{|\mathcal{E}|}{2} - \deg g_{1,1} && \deg p_2 \leq \tfrac{|\mathcal{E}|}{2} - 1 - \deg g_{2,2} && \text{if } |\mathcal{E}| \text{ is even} \\ &\deg p_1 \leq \tfrac{|\mathcal{E}|-1}{2} - \deg g_{1,1} && \deg p_2 = \tfrac{|\mathcal{E}|-1}{2} - \deg g_{2,2} && \text{if } |\mathcal{E}| \text{ is odd} \end{aligned}$$

Furthermore, let $h_1(x) = g_{1,1}^2 + x g_{1,2}^2$ and $h_2(x) = g_{2,1}^2 + x g_{2,2}^2$ and define $(y_i : z_i) = \left(\sqrt{h_1(\alpha_i)} : -\sqrt{h_2(\alpha_i)}\right)$ for $i = 1, \ldots, n$. Then for exactly $|\mathcal{E}|$ out of the n choices of i, we have

$$z_i p_1(\alpha_i) - y_i p_2(\alpha_i) = 0 \tag{5.8}$$

Proof. We first show the degree bounds on p_1 and p_2. Assume first that $|\mathcal{E}|$ is even; then $\deg a = |\mathcal{E}|/2$ and $\deg b \leq |\mathcal{E}|/2 - 1$ so $\text{LP}(a, b) = 1$. Therefore Proposition 2.14 on page 17 states exactly the given degree bounds. Similarly, if $|\mathcal{E}|$ is odd, then $\deg a \leq |\mathcal{E} - 1|/2$ and $\deg b = (|\mathcal{E}| - 1)/2$, so $\text{LP}(a, b) = 2$ and employing the lemma again gives the sought.

Now if $|\mathcal{E}| \leq \deg G$ and since $\deg g_{1,1} + \deg g_{2,2} = \deg G$ then $\deg p_1 + \deg p_2 < 0$ by the above, which implies that $p_1 = 0$ or $p_2 = 0$. It must therefore be so that $(a, b) = p_i \boldsymbol{g}_i$ for i being 1 or 2, and $p_{3-i} = 0$. We know that $\Lambda = a^2 + xb^2$, so if $\gcd(a, b) > 1$ then $\gcd(a, b)^2 \mid \Lambda$; but Λ by definition is square-free. Thus $\gcd(a, b) = 1$ implying that $\deg p_i = 0$.

Assume now that $|\mathcal{E}| > \deg G$. Since we just argued that $\gcd(a, b) = 1$, then clearly $\gcd(p_1, p_2) = 1$. Now

$$\begin{aligned}\Lambda(x) = a^2 + xb^2 &= (p_1 g_{1,1} + p_2 g_{2,1})^2 + x(p_1 g_{1,2} + p_2 g_{2,2})^2 \\ &= p_1^2(g_{1,1}^2 + x g_{1,2}^2) + p_2^2(g_{2,1}^2 + x g_{2,2}^2)\end{aligned}$$

For each $i \in \mathcal{E}$ we therefore have

$$0 = \Lambda(\alpha_i) = p_1(\alpha_i)^2 h_1(\alpha_i) + p_2^2 h_2(\alpha_i)$$

But again, since we are in characteristic 2, the square root exists and distributes over addition, so:

$$p_1(\alpha_i)\sqrt{h_1(\alpha_i)} + p_2(\alpha_i)\sqrt{h_2(\alpha_i)} = 0$$

As before, for all $i \notin \mathcal{E}$, Λ evaluates non-zero, which is still left non-zero after drawing square-roots, so the above equation will not hold. Therefore, it is indeed exactly $|\mathcal{E}|$ of the defined points that p_1, p_2 interpolates.

The last step is to show that the defined points are indeed projective, i.e. h_1 and h_2 do not have a common zero in L. There must be some $U \in \mathbb{F}[x]^{2\times 2}$ with determinant 1 such that $\left[\frac{\boldsymbol{g}_1}{\boldsymbol{g}_2}\right] = U M_{\text{Wu}}^G$. Squaring is a linear operation in $\mathbb{F}_{2^m}[x]$, so we also get

$$\begin{pmatrix} g_{1,1}^2 & g_{1,2}^2 \\ g_{2,1}^2 & g_{2,2}^2 \end{pmatrix} = U^{(2)} \begin{pmatrix} 1 & \tilde{S}^2 \\ 0 & G^2 \end{pmatrix}$$

where $U^{(2)}$ is the component-wise square of U. Note that we still have $\det U^{(2)} = 1$. Therefore

$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} g_{1,1}^2 & g_{1,2}^2 \\ g_{2,1}^2 & g_{2,2}^2 \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = U^{(2)} \begin{pmatrix} 1 & \tilde{S}^2 \\ 0 & G^2 \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = U^{(2)} \begin{pmatrix} 1 + x\tilde{S}^2 \\ xG^2 \end{pmatrix}$$

Since $\det U^{(2)} = 1$, then h_1, h_2 can only have a common root if $1 + x\tilde{S}^2$ and xG^2 have the same common root. But clearly 0 is not a root of $1 + x\tilde{S}^2$, and xG^2 can have no other roots in $\mathbb{F}_{2^m}$. □

The case $|\mathcal{E}| \leq \deg G$ is exactly Patterson's decoder using the language of module minimisation. For $|\mathcal{E}| > \deg G$, we then have a rational interpolation problem if we knew the value of $|\mathcal{E}|$ in advance. Since we do not, we will again rely on Lemma 5.8 on page 132, allowing us to choose the parameters of the interpolation problem based on a worst case $|\mathcal{E}| = \tau$, while still ensuring that solutions will be found for $|\mathcal{E}| < \tau$. The parameter choices are complicated by the distinction between an even and an odd number of errors. For using Lemma 5.8, we are forced to choosing θ_1 and θ_2 such that the distance from the degree upper bounds on p_1, p_2 increase gradually with a decreasing number of errors; hence:

$$\theta_1 = \tfrac{\tau}{2} - \deg g_{1,1} \qquad \theta_2 = \tfrac{\tau-1}{2} - \deg g_{2,2} \qquad \text{and so} \qquad \theta = \tau - \tfrac{1}{2} - \deg G$$

since $\deg g_{1,1} + \deg g_{2,2} = \deg \det M^G_{\text{Wu}} = \deg G$. Note that either θ_1 or θ_2 as well as θ are not integers. With these choices, how high can we then choose τ? By Proposition 5.7 on page 131, our only restriction is $\tau^2 > n\theta$, i.e.

$$\tau^2 > n(\tau - \tfrac{1}{2} - \deg G) \qquad \Longleftrightarrow \qquad \tau < \tfrac{1}{2}n - \tfrac{1}{2}\sqrt{n(n - 4\deg G - 2)}$$

This upper bound is exactly $J_2(n, 2\deg G + 1)$, i.e. the binary Johnson bound for the designed minimum distance $2\deg G + 1$, which was discussed in Section 3.6.

Collecting the steps of the decoder yields Algorithm 7. We have

Theorem 5.28. *Algorithm 7 is correct.*

Proof. The algorithm is very close to Algorithm 6 on page 143, so the proof of correctness is as well. In Line 3 we break if the module minimisation finds an error locator *and* the minimum distance of the code, $2\deg G + 1$, dictates that there can be no other words within τ.

Constructability of Q in Line 4 follows from the above discussion and that suitable parameter choices exists by Proposition 5.7 on page 131. Note that by that proposition we can indeed choose s and ℓ such that $2s \leq \ell$ since $\frac{\tau}{\theta} = \frac{\tau}{\tau - 1/2 - \deg G}$ which is at least 2 whenever $\tau < 2\deg G + 1$.

By Proposition 5.27 on page 149 it must therefore be the case that when $|\mathcal{E}| = \tau$, we find p_1, p_2 among the $p_1^\star, p_2^\star$, and that Λ is therefore found in Line 6. Whenever $|\mathcal{E}| < \tau$, this will also be the case by Lemma 5.8 on page 132 as long as

$$\min\{\theta_1 - \deg p_1,\ \theta_2 - \deg p_2\} \geq \tfrac{s}{\ell}(\tau - |\mathcal{E}|)$$

But $\deg p_1 \leq \frac{|\mathcal{E}|}{2} - \deg g_{1,1}$ and $\deg p_2 \leq \frac{|\mathcal{E}|-1}{2} - \deg g_{2,2}$ by Proposition 5.27, and so $\theta_1 - \deg p_1 \geq \frac{\tau}{2} - \frac{|\mathcal{E}|}{2}$, and similarly $\theta_2 - \deg p_2 \geq \frac{\tau}{2} - \frac{|\mathcal{E}|}{2}$. Since $\frac{s}{\ell} \leq 2$ the requirement is therefore satisfied. □

The asymptotic complexity of Algorithm 7 is clearly dominated by the solving the rational interpolation problem; thus the complexities of Table 5.2 on page 144 are carried over unchanged.

[6] In [BHNW13, p.3278], these degree bounds erroneously read "less than".

Algorithm 7 Wu list decoding binary Goppa codes

Input: The received word $\boldsymbol{r} = (r_1, \dots, r_n)$. Parameters s, ℓ, τ such that $\tau < \frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\deg G - 2)}$ and $E_{\text{Wu}}^{[n,\tau-1/2-\deg G]}(s, \ell, \tau) > 0$ as well as $2s \leq \ell$ for the given code, e.g. by using Proposition 5.7 on page 131.

Output: A list of all codewords in $\mathcal{C}$ within radius τ of $\boldsymbol{r}$ or Fail if there are no such.

1 Calculate the syndrome $S(x)$ from r according to (5.5) on page 147. If $S^{-1}(x) = x$ and $0 \in L$, then flip the corresponding bit of r and return that word. If $S^{-1}(x) = x$ and $0 \notin L$, return Fail. Otherwise, calculate $\tilde{S}(x)$ satisfying $\deg \tilde{S} < \deg G$ and $\tilde{S}^2(x) \equiv x + S^{-1}(x) \mod G(x)$.

2 Compute $[\frac{\boldsymbol{g}_1}{\boldsymbol{g}_2}]$ by module minimising M_{Wu}^G of (5.7) on page 149, with $\text{LP}(\boldsymbol{g}_1) = 1$.

3 If either $g_{1,1}$ or $g_{2,1}$ is a valid error-locator of degree at most $2\deg G - \tau$, use it to correct $\boldsymbol{r}$, and if this yields a word in $\mathcal{C}$, return this one word.

4 Otherwise, set $\theta_1 = \frac{\tau}{2} - \deg g_{1,1}$ and $\theta_2 = \frac{\tau-1}{2} - \deg g_{2,2}$. Construct a $Q(x, y : z)$ satisfying the requirements of Theorem 5.4 on page 130 using the points $\left\{\left(\alpha_i, \sqrt{h_1(\alpha_i)} : -\sqrt{h_2(\alpha_i)}\right)\right\}_{i=0}^{n-1}$, where h_1, h_2 are as in Proposition 5.27 on page 149, and using the parameters τ, s, ℓ.

5 Find all $p_1^\star, p_2^\star$ such that[6] $\deg p_1^\star \leq \theta_1$ and $\deg p_2 \leq \theta_2$ and $Q(x, p_1^\star : p_2^\star) = 0$. Return Fail if no such factors exist.

6 For each such factor, construct $\Lambda^\star(x) = {p_1^\star}^2 h_1 + {p_2^\star}^2 h_2$. If it is a valid error-locator, use it for correcting $\boldsymbol{r}$. Return Fail if none of the factors yield error-locators.

7 Return those of the corrected words that are in $\mathcal{C}$. Return Fail if there are no such words.

Example 5.29. *Consider a binary Goppa code of length* $n = 128$ *over* $\mathbb{F}_{128}$ *with* $\deg G = 14$*, i.e. dimension at least* $n - 7\deg G = 30$*. This has minimum distance at least* 29*, so Patterson's decoder can correct* 14 *errors. Using the Wu list decoder, we can correct up to* 16 *errors, so set* $\tau = 16$*. This gives* $\theta = 3/2$ *and from Proposition 5.7 on page 131 we can choose* $(s, \ell) = (3, 26)$*; this is also the minimal possible. The resulting matrix to module minimise is therefore* 27×27*; assuming* $z_i = 0$ *for 5 values of* i *then in a typical run the weighted matrix* D *to be minimised would have* $\text{maxdeg}\, D = 405$*,* $\deg D = 10746$ *and* $\Delta(D) = 9458$*.* ♠

Remark. Similar to the case with GRS codes, remarked upon on page 143, if we are decoding a single error beyond $\deg G$ then we will have $\theta = \frac{1}{2}$, and the rational interpolation step will degenerate and can be performed much faster: since then both $\deg p_1 < 1$ and $\deg p_2 < 1$, we are, in effect, searching for *constants* $p_1, p_2 \in \mathbb{F}_{2^m}$, and this means that all the $(y_i : z_i)$ for $i \in \mathcal{E}$ are equivalent. We therefore simply group all the $(y_i : z_i)$ into equivalence classes, and each such group gives rise to a possible error locator. We can prune this list and remove all the small groups in the same manner as we did for GRS codes, so we will be considering only around $\lfloor \frac{n}{\tau} \rfloor$ error locators in the worst case. ♦

Remark. It is important to keep in mind the asymptotics for the case of binary

Goppa codes. The degree of the extension field m must satisfy $m \geq \log_2 n$. To be sure that we do not end up with a trivial code, we a priori need to choose G such that $m \deg G < n$, and therefore, we cannot keep $\deg G/n$ a constant, as we can d/n for GRS codes. Our decoding radius $\frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\deg G - 2)}$ for large n therefore becomes at most $\frac{1}{2}n(1 - \sqrt{1 - \frac{\log_2 n}{n}}) \in o(n)$. So we are claiming a polynomial time decoder in n which decodes sub-linearly many errors; with n growing, the "relative expense" of decoding errors therefore increases. Aside from the calculation of the syndrome, the complexity of Patterson's minimum distance decoder (i.e. minimising M^{Γ}_{Wu}, (5.7) on page 149) is linear in $\deg G$, the decoding radius; it would be nice if it was similarly possible to bound the list decoding complexity by $\deg G$.

A more concrete observation is that often, and for useful parameters of Goppa codes, $J_2(n, 2\deg G + 1)$ is only one or two errors beyond $\deg G$. Since the necessary values of s and ℓ can become rather large when decoding close to the bound, and since our field is $\mathbb{F}_2$, it might in concrete cases be a much faster strategy to *chase* these few extra errors: i.e. randomly guess a few error positions, flip these, and use a shorter-range decoder afterwards. If that fails, guess different positions. In combination with the plus-one decoder remarked upon above, one could for certain code lengths easily imagine that this could decode e.g. two errors beyond $\deg G$ cheaper than using the full list decoder. Especially in hardware, this is intriguing, since chasing is easy to implement, while the full module minimisation and root-finding step requires comparatively large amounts of logic and memory. More in-depth simulation, and possibly analysis using asymptotic expansion of the leading term, is required to say more on for which parameters this strategy is better. See also the next section. ◆

5.3.1 Parallel decoding beyond the binary Johnson bound

One can actually easily do slightly better, and decode just beyond the binary Johnson bound, by noting a sub-optimality in the choice of θ above, and by allowing two parallel runs of the rational interpolation procedure. The key lies in treating two cases depending on the parity of $|\mathcal{E}|$, just like the upper bounds on the degrees of p_1 and p_2 do in Proposition 5.27 on page 149. Specifically:

Lemma 5.30. *In the context of Proposition 5.27 for the case $|\mathcal{E}| > \deg G$, let $\tau^{(0)}$ be an even number. Choose*

$$\theta_1^{(0)} = \tfrac{\tau^{(0)}}{2} - \deg g_{1,1} \qquad \theta_2^{(0)} = \tfrac{\tau^{(0)}}{2} - 1 - \deg g_{2,2}$$

If $|\mathcal{E}| \leq \tau^{(0)}$ and is even, and if $(\tau^{(0)})^2 > n(\theta_1^{(0)} + \theta_2^{(0)})$, then there exists parameters s, ℓ with $2s \leq \ell$ and such that any $Q \in \mathbb{F}[x][y:z]^{\ell}$ satisfying the requirements of Theorem 5.4 on page 130 will also satisfy $Q(x, p_1(x), p_2(x)) = 0$.

Similarly, let $\tau^{(1)}$ be an odd number. Choose

$$\theta_1^{(1)} = \tfrac{\tau^{(1)}-1}{2} - \deg g_{1,1} \qquad \theta_2^{(1)} = \tfrac{\tau^{(1)}-1}{2} - \deg g_{2,2}$$

If $|\mathcal{E}| \leq \tau^{(1)}$ and is odd, and if $(\tau^{(1)})^2 > n(\theta_1^{(1)} + \theta_2^{(1)})$, then the same statement holds.

Proof. Consider first $\tau^{(0)}$. Then let $\theta^{(0)} = \theta_1^{(0)} + \theta_2^{(0)} = \tau^{(0)} - 1 - \deg G$ since $\deg g_{1,1} + \deg g_{2,2} = \deg \det M_{\text{Wu}}^G = \deg G$. As $\tau^{(0)} < 2\deg G + 1$ then we see $\frac{\tau^{(0)}}{\theta^{(0)}} \geq 2$, so if $(\tau^{(0)})^2 > n\theta^{(0)}$ then by Proposition 5.7 on page 131, we can choose parameters such that $E_{Wu}(s, \ell, \tau^{(0)}) > 0$ and $2s \leq \ell$. Thus, if $|\mathcal{E}| = \tau^{(0)}$, Theorem 5.4 immediately implies that $Q(x, p_1(x) : p_2(x)) = 0$. As usual, for $|\mathcal{E}| < \tau^{(0)}$, we employ Lemma 5.8 on page 132. By this, $Q(x, p_1(x) : p_2(x)) = 0$ as long as

$$\min\{\theta_1^{(0)} - \deg p_1,\ \theta_2^{(0)} - \deg p_2\} \geq \tfrac{s}{\ell}(\tau - |\mathcal{E}|)$$

But if $|\mathcal{E}|$ is even, then by Proposition 5.27, we have $\theta_1^{(0)} - \deg p_1 = \frac{\tau^{(0)}}{2} - \deg g_{1,1} - (\frac{|\mathcal{E}|}{2} - \deg g_{1,1}) = \frac{\tau^{(0)} - |\mathcal{E}|}{2}$, and similarly, $\theta_2^{(0)} - \deg p_2 \geq \frac{\tau^{(0)} - |\mathcal{E}|}{2}$. Since $\frac{s}{\ell} \leq \frac{1}{2}$, the requirement is satisfied. The same series of arguments proves the case for $\tau^{(1)}$. □

The important observation is now that in both the even and odd case above, $\theta^{(i)} = \tau_i - 1 - \deg G$, i.e. $\frac{1}{2}$ less than in the previous section. This leads to

Proposition 5.31. *We can modify Algorithm 7 such that in Line 4, we instead construct two interpolation polynomials, $Q^{(0)}$ and $Q^{(1)}$, corresponding to the choices of $\theta_1^{(i)}, \theta_2^{(i)}$ given in Lemma 5.30 for $\tau^{(0)}, \tau^{(1)}$ chosen as the greatest integer at most τ which is even, respectively odd. The parameters s, ℓ are chosen individually for the two runs according to $\theta^{(i)}$. In Line 5, we then instead find all $p_1^\star, p_2^\star$ which satisfy either $Q^{(0)}(x, p_1^\star : p_2^\star) = 0$ or $Q^{(1)}(x, p_1^\star : p_2^\star) = 0$.*

The algorithm is still correct and allows to choose any $\tau < \frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\deg G - 4)}$.

Proof. From Lemma 5.30 follows all the claims except the upper bound on the choice of τ. Since in both the construction of Q_0 and Q_1, we have $\theta^{(i)} = \tau_i - 1 - \deg G$, for $i = 0, 1$ both $\tau^{(0)}$ and $\tau^{(1)}$ are bounded by having to satisfy

$$\tau_i^2 > n(\tau_i - 1 - \deg G) \qquad \Longleftrightarrow \qquad \tau_i < \tfrac{1}{2}n - \tfrac{1}{2}\sqrt{n(n - 4\deg G - 4)}$$

Thus, choosing any τ less than the above, and then $\tau_i \leq \tau$ for $i = 0, 1$ must be satisfactory. □

Obviously, the asymptotic running time of this algorithm is intact, since we are simply running two instances of rational interpolation instead of one. Thus again, the computational complexities from Table 5.2 on page 144 are carried over.

Example 5.32. *Consider the* [128, ≥30, ≥29] *code from Example 5.29. Using the parallel decoder, we can decode this up to $\tau = \tau^{(1)} = 17$ errors with $\tau^{(0)} = 16$. This gives $\theta^{(0)} = 1$ and $\theta^{(1)} = 2$, which leads to us choosing parameters $(s^{(0)}, \ell^{(0)}) = (1, 12)$ and $(s^{(1)}, \ell^{(1)}) = (7, 56)$ by using Proposition 5.7 on page 131. As expected, correcting only the even errors up to 16 is quite a lot cheaper than correcting the odd ones up to 17.*

The method can, however, also be an advantage even though we do not decode further than the binary Johnson bound. In the same example, we could choose to decode up to the binary Johnson bound and choose $\tau = \tau^{(0)} = 16$ *leading to* $\tau^{(1)} = 15$*. This gives* $\theta^{(0)} = 1$ *and* $\theta^{(1)} = 0$*, which leads us to choosing parameters* $(s^{(0)}, \ell^{(0)}) = (1, 12)$ *and* $(s^{(1)}, \ell^{(1)}) = (1, 8)$*. Minimising the two resulting matrices and root-finding in the two found Q-polynomials should be much faster than performing only one pass with the higher parameters* $(s, \ell) = (3, 26)$ *which were necessary in Example 5.29. Even better, since* $\theta^{(1)} = 0$*, we can in the odd errors-pass use the degenerate version of rational interpolation remarked upon on page 152 and therefore perform this rational interpolation very quickly.* ♠

Remark. $\frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\deg G - 4)}$ is not much bigger than $J_2(n, 2\deg G + 1) = \frac{1}{2}n - \frac{1}{2}\sqrt{n(n - 4\deg G - 2)}$ and asymptotically not at all. However, it is a few errors more, and for rather short codes—and the designed minimum distance of binary Goppa codes are only really good when the codes are not too long—this is not insignificant. As the above example shows, it can also significantly reduce the complexity of the decoding up to the binary Johnson bound. One should also again keep in mind the remark on asymptotic behaviour from page 152. ♦

5.3.2 Relation to Guruswami–Sudan decoding of Alternant codes

As mentioned in Section 3.6, Kötter and Vardy described how the Guruswami–Sudan algorithm can be made to decode any Alternant code up to the small-field Johnson bound. More precisely, an Alternant code over $\mathbb{F}_q$ is a sub-field sub-code of some GRS code over, say, $\mathbb{F}_{q^m}$. Therefore, the Guruswami–Sudan algorithm can trivially be applied to decode the Alternant code up to $J(n, d)$, i.e. the Johnson bound of the GRS code's minimum distance d. However, this can be improved by integrating into the method, through a clever choice of multiplicities, that we are searching for a codeword in $\mathbb{F}_q$ and not $\mathbb{F}_{q^m}$: instead of choosing just one multiplicity s for all (α_i, r'_i), one additionally chooses a lower multiplicity $\hat{s}$ for each of the points (α_i, γ) for $\gamma \in \mathbb{F}_q \setminus \{r'_i\}$. Choosing all $s, \hat{s}, \ell$ in the right way, this turns out to increase the decoding radius to $J_q(n, d)$, i.e. the q-ary Johnson bound mentioned in Section 3.6. We'll call this method GS+KV; this was circulated in a preprint of [KV03a] but removed in the published version; it is described by Roth [Rot06, Section 9.6] and for the extension to Goppa codes mentioned below by Augot, Barbier and Couvreur [ABC10].

Any Goppa code is also an Alternant code, see e.g. [MS77], so we can use the above to decode up to $J_q(n, d)$. However, sometimes we know that the Goppa code has better minimum distance than d, in particular for binary Goppa codes. This good minimum distance is usually a consequence of an elegant result by Sugiyama et al. [SKHN76],

which equates Goppa codes defined by two different Goppa polynomials:

$$\Gamma\Big(\prod_i G_i^{a_i}, L\Big) = \Gamma\Big(\prod_i G_i^{a_i+b_i}, L\Big) \qquad b_i = \begin{cases} 1 & \text{if } a_i \equiv q-1 \mod q \\ 0 & \text{otherwise} \end{cases}$$

Here, the G_i are distinct, irreducible polynomials, and q is the size of the field. In particular, if $q = 2$ and G is square-free then $\Gamma(G, L) = \Gamma(G^2, L)$. Using the right-hand side code, we are told that the left-hand side code has basically twice the minimum distance we would expect from looking at its surrounding GRS code. This fact is what we stated in the beginning of Section 5.3 when we claimed that the minimum distance was $2\deg G + 1$ (and not $\deg G + 1$)[7].

Naturally, we are interesting in decoding such a Goppa code up to $J_2(n, 2\deg G+1)$ and not just $J_2(n, \deg G + 1)$. This we saw how to do using Wu list decoding in Section 5.3, but using the above identity on Goppa codes, we have another method: just use the GS+KV on the GRS code surrounding the Goppa code $\Gamma(G^2, L)$!

Intriguingly, it turns out that the parameters of this method are related to the parameters of the Wu method in a manner analogous to the relation of Section 5.2.2. Specifically, it is easy to show in the same manner as in Proposition 3.4, that the parameters $s, \hat{s}, \ell, \tau$ are valid choices for the GS+KV whenever $E^{[n,k]}_{\text{GS+KV}}(s, \hat{s}, \ell, \tau) > 0$ where

$$E^{[n,k]}_{\text{GS+KV}}(s, \hat{s}, \ell, \tau) = (\ell+1)(s(n-\tau)+\hat{s}\tau) - \tbinom{\ell+1}{2}(k-1) - \tbinom{s+1}{2}n - \tbinom{\hat{s}+1}{2}n(q-1)$$

We then have

Proposition 5.33. *$E^{[n,n-2\deg G]}_{\text{GS+KV}}(\ell - s, s, \ell, \tau) = 2E^{[n,\tau-1/2-\deg G]}_{\text{Wu}}(s, \ell, \tau)$ when $q = 2$.*

Proof. By insertion into Definition 5.5 on page 130 of $E^{[n,\theta]}_{\text{Wu}}$ and some rewriting. □

Thus, if we can Wu list decode τ errors in a binary Goppa code with Goppa polynomial G, then we can GS+KV list decode τ errors in a binary Alternant code with minimum distance $2\deg G + 1$. Incidentally, $\Gamma(G^2, L)$ is exactly such an Alternant code. This also immediately leads to lower bounding the decoding radius of the GS+KV in the binary case:

Corollary 5.34. *The GS+KV can decode an $[n, \geq n - m(d-1), d]$ Alternant code over $\mathbb{F}_2$ with designed minimum distance d and $n \leq 2^m$ up to $J_2(n, d) = \frac{1}{2}n - \frac{1}{2}\sqrt{n(n-2d)}$.*

Since the relation of Proposition 5.33 restricts the parameters of GS+KV to $s+\hat{s} = \ell$, we cannot from this vantage point be sure that there are not better choices of $s, \hat{s}, \ell$ leading to greater decoding radii; it turns out one cannot do better, but for that we need to analyse $E^{[n,k]}_{\text{GS+KV}}$ directly, see e.g. [Rot06, Section 9.6 and Problem 9.9].

[7] This high minimum distance of course also follows from Patterson's rewrite: we could not perform unique decoding with (5.5) on page 148 up to $\deg G$ errors if the minimum distance was not at least $2\deg G + 1$.

Of course, the relation also does not prove that the decoding radius of GS+KV for $q > 2$ is $J_q(n, d)$, but this is shown in the same reference.

Proposition 5.33 also implies closed form expressions for the parameters of the GS+KV for achieving this decoding radius, through the parameters of Proposition 5.7 on page 131. Again, since we are restricting ourselves to $s + \hat{s} = \ell$, we can't immediately be sure that the near optimality of the parameter choices of Proposition 5.7 on page 131 carries over, since other relative choices of $s, \hat{s}, \ell$ might be smaller.

Example 5.35. *From Proposition 5.33, it follows that the* $[128, \geq 30, \geq 29]$ *code from Example 5.29 on page 152 can be GS+KV decoded up to* 16 *errors, and that we can choose* $(s, \hat{s}, \ell) = (23, 3, 26)$*. As discussed in Section 3.6, Lee and O'Sullivan [LO06] gave a method to compute a basis for a module in which resides a satisfactory interpolation polynomial, but I have not looked deeply into how this basis turns out and how its key properties are, so unfortunately I can not compare with the module we had to minimise for Example 5.29, described in Example 5.17 on page 139. As an alternative, we can compare the size of the linear system of equations which would have to be solved, if we were to use Gaussian elimination for finding* Q*, by picking out the relevant terms of* $E_{\text{Wu}}^{[n,\tau-1/2-\deg G]}(s, \ell, \tau)$ *respectively* $E_{\text{GS+KV}}^{[n,n-2\deg G]}(s, \hat{s}, \ell, \tau)$*: for Wu list decoding, we would get* 768 *equations in* 770 *unknowns, while for GS+KV we would get* 36 096 *equations in* 36 099 *unknowns. Note that the number of unknowns is the number of coefficients in the respective* Q *polynomials, so the size of this also plays a role in the root-finding step.* ♠

5.4 Related work

The Wu list decoder is relatively recent, so the amount of related work is limited. The decoder was proposed for GRS and binary BCH codes in [Wu08], but the first preprint of the article already circulated in early 2007. It is still essentially the only alternative to Guruswami–Sudan for decoding GRS codes up to $J(n, d)$.

Wu developed the algorithm as an extension to the Berlekamp–Massey algorithm applied to the syndrome key equation. For us in Section 5.2, it was easy to continue after the analogous computation of the module minimisation, since we had a Gröbner basis; however, Wu basically has to show all the properties such a basis entails, in a less transparent setting, for the Berlekamp–Massey. My own master's thesis [Nie10] was a more verbose exposition of this approach, and for full, clear proofs of all necessary properties, the algorithm explained from end to end becomes lengthy and very detailed; indeed, one gets the feeling that had a single detail in the underlying algebra been different, the algorithm could not work. I hope that the exposition given here does not invoke the same feeling.

Trifonov was the first to bring in the less fragile, more flexible language of Gröbner bases [Tri10a]. Instead of using a key equation, he develops the algorithm from the Welch–Berlekamp (see e.g. Section 4.3.1); the resulting module to minimise is

still M^{Γ}_{Wu} of (5.2) on page 140, however. Trifonov also described Q as a trivariate polynomial, where Wu had originally used a less algebraically transparent language of bivariate polynomials whose second variable is allowed to take the value ∞. This allowed Trifonov to extend his binary exponentiation interpolation algorithm discussed in Section 3.6 to handle the rational interpolation case. That discussion carries over to this extended version, in particular with regards to the complexity. Trifonov also performed an analysis of the parameter choices for the rational interpolation; in [Tri10a], however, there was a mistake in that he requested one to pick ℓ from an interval which was not guaranteed to hold any integers. This was corrected in the, otherwise almost equivalent, journal version [TL12] after having been pointed out by a reviewer. The resulting choice of parameters is equivalent to ours in Proposition 5.7 on page 131.

About a year later than Trifonov's conference contribution, Ali and Kuijper published a journal article [AK11] containing some of the same ideas; in particular obtaining a Gröbner basis for the Welch–Berlekamp module and continuing the Wu approach on this. This article has a number of issues I feel obliged to point out:

- They completely disregard that there might be points at infinity, so their description of the rational interpolation step is not sound. They do not use the trivariate polynomials of Trifonov.
- They consider their algorithm a "minimal list decoder" since it, like our multi-trial decoder in Section 3.3, only finds the list of *nearest* codewords; however, they turn the usual Wu list decoder into one such in a trivial manner: begin by half-the-minimum distance decoding and expand the decoding radius gradually, running the whole decoding algorithm each iteration. The only computation they save from one iteration to the next is the computation of a Gröbner basis of M_{Wu}; however, the price for this computation is completely overshadowed by that of just a single instance of rational interpolation. But in their complexity analysis, at the very end of the paper, they *forget* the factor $\tau - d/2$ for running that many instances of the interpolation algorithm!
- They analyse the parameter choices and find a minimal s for which there exists a valid choice of ℓ (finding the same as we in Proposition 5.7, though specialised for decoding of GRS codes), but instead of choosing the minimal accompanying ℓ, they search for one which minimises the quantity ℓT, where T is the number of coefficients in Q. The rationale is that this measure ℓT is dominating in the *memory* complexity of the Kötter interpolation method, which they use (that algorithm was also discussed in Section 3.6). However, that memory complexity estimate is strictly asymptotic, and since their found ℓT is not asymptotically better than e.g. from choosing our ℓ (apparently), it is completely unclear whether there is benefit or penalty memory-wise in seeking minimal ℓT instead of minimal ℓ, and furthermore what impact this has on the computational complexity.

It should also be mentioned that neither Trifonov nor Ali and Kuijper recognises the need for Lemma 5.8 on page 132: i.e. that it is not immediately clear that in a

decoding instance where fewer than τ errors have occurred, the rational interpolation procedure still finds the sought coefficient polynomials p_1, p_2. This is an important complication: indeed, this was the root of the surprising non-integer choice of θ_1, θ_2 in the Wu list decoder for binary Goppa codes in Section 5.3.

Wu himself proposed closed expressions for choosing s, ℓ in [Wu08], and analysed how these compared to that of the GS. The widespread belief that "Wu's list decoder needs lower multiplicity than Guruswami–Sudan" comes from this comparison. However, this comparison is quite unfair: it compares the asymptotically satisfactory, but clearly suboptimal, parameters described by Guruswami and Sudan in [GS99] with those of Wu; but Wu had simply analysed the governing equation for his list decoder a bit more tightly and then easily came out on top in this comparison. The true comparison is as we showed in Section 5.2.2: ℓ can always be chosen the same, and the multiplicities are reflected around $\ell/2$; for $\tau < n/2$, Wu requires the lower s, while it requires the higher s for $\tau > n/2$. This is also usually in the Wu list decoder's favour, since practical applications are rarely working over channels that corrupt more than half the signal!

Wu also described in [Wu08] how to use his paradigm to list decode BCH codes up to $J_2(n, d) = \frac{1}{2}\big(n - \sqrt{n(n-2d)}\big)$, i.e. the binary Johnson bound for the designed minimum distance d. The method utilises a simple relation in the syndrome polynomials of BCH codes, enabling one to construct a syndrome polynomial of basically twice the degree. I have not in depth examined how this translates to the Gröbner basis description of the key equation, but I expect no conceptual difficulties with such an adaption.

Bernstein [Ber11b] seem to have described an algorithm almost identical to the Wu list decoder applied to binary Goppa codes, but coming from the Coppersmith–Howgrave-Graham language (see Section 3.6). He also explains how Λ can be obtained as a small linear combination of the rows of a minimised M_{Wu}^{Γ} (from (5.7) on page 149), and he then uses the function field analogue of Coppersmith and Howgrave-Graham's technique to obtain this linear combination. However, for some reason he does not obtain the same list decoding radius, but only $J(n, 2\deg G + 1) = n - \sqrt{n(n - 2\deg G - 2)}$; this is the same radius one achieves if decoding the GRS code surrounding $\Gamma(G^2, L)$ using the Guruswami–Sudan (and not using the Kötter–Vardy multiplicity assignment). See e.g. Figure 5.2 on page 148. I have not studied the paper well enough to be certain why his decoding radius is inferior, but it seems to be due to the setup: in the way he invokes Coppersmith–Howgrave-Graham, he is in effect looking for $g_{1,1}^2/g_{2,1}^2$, and not, as we $g_{1,1}/g_{2,1}$. It is possible he is simply missing the straightforward square-rooting that we performed towards the end of the proof of Proposition 5.27 on page 149.

Shortly afterwards, Bernstein published another decoding method [Ber11a]; this was already mentioned in Section 3.6 since it is essentially the GS+KV, where Q is found by module minimisation of an explicit basis. Again, he is coming from the Coppersmith–Howgrave-Graham language. As Bernstein himself points out,

and by the discussion in Section 5.3.2, this can decode a binary Goppa code up to $J_2(n, 2 \deg G + 1)$.

Chapter 6

Conclusion

The possibility of decoding algebraic codes, and in particular GRS codes, beyond half the minimum distance is one of the great advancement in the field within the last two decades. For practical applications in the traditional coding setting, there are still some stumbling blocks, primarily the increased computational complexity of executing them compared to classical key equation decoding.

In this thesis, we have investigated how three decoding paradigms—the Guruswami–Sudan, Power decoding and the Wu algorithm—at their computational core need to solve certain $\mathbb{F}[x]$-linear problems, and how this can be accomplished using module minimisation. The main conclusions are that using the fastest techniques in the literature for module minimisation, we can meet or improve the fastest, previously known realisations of each of the decoding paradigms. Moreover, it seems that many previous approaches are computationally equivalent to ours when applying the Mulders–Storjohann or Alekhnovich module minimisation algorithms. This work therefore unifies and provides a better foothold of a long history of algebraic decoding algorithms, encompassing key equation solving by Berlekamp [Ber68] up to Gröbner basis approaches to decoding Hermitian codes [LO09], along with several completely new decoders in between.

As we have seen, module minimisation is a powerful technique, and it has been applied throughout this thesis with repetitive simplicity. The divide between the modelling of the decoding problem and the computational algebra for performing the module minimisation makes the methods conceptually easy to understand, and has proven to be highly effective in attaining the best known complexities. However, there are limitations to module minimisation; in particular, there are good reasons to

believe that the worst-case complexity of the Giorgi–Jeannerod–Villard algorithm in the pure setting, and with $n \gg m$ can not really be improved, since module minimisation can not be performed faster than computing the determinant [GJV03]. Therefore, if we wish to further improve the worst-case complexity of *any* of the three studied decoding paradigms, we need to seek outside the realm of algorithms which are roughly equivalent to module minimisation; in particular, I believe this includes more or less all known generalisations of the Berlekamp–Massey and the Euclidean algorithm. This is a key insight: the often intricate and complicated generalisations of these two classical algorithms seem in all cases I have studied to be closely describable as one of the module minimisation algorithms presented here; so whenever the urge arise to create yet another generalisation of either of these, one should first be convinced that the problem is not easily solvable by module minimisation, in which case an asymptotic benefit is likely not attainable. Sakata's algorithm is possibly a prominent exception to this, see the discussion in the next section.

The thesis has also touched upon many other theoretical and practical aspects of algebraic coding, in particular generalisations of the paradigms to other codes. We will close the thesis with a discussion of many extensions of the work which it could be interesting to pursue.

6.1 Future directions

The following discussion is divided into three subsections, roughly based on how difficult the extensions would be to carry out. Many of these have already been mentioned earlier in the thesis in remarks or in the Related work sections.

Polishing the results

These are a few obvious improvements to the results which could easily have been part of this thesis, but which I, primarily due to lack of time, did not complete.

- For a 2D key equation with $\sigma = 1, \nu = 1$ and $G_j = x^{n_j}$ for some $n_j \in \mathbb{N}_0$, then Roth and Ruckenstein's generalisation the Berlekamp–Massey [RR00] can be applied, and its complexity estimate is significantly better than that of the Demand–Driven algorithm; see discussion in Section 2.7. I am quite certain that the two methods here are computationally, step-for-step equivalent, which means that it is our analysis of the Demand–Driven algorithm which is lacking. A similar discrepancy is present for higher σ between the Zeh–Gentner–Augot algorithm and the Demand–Driven.
- It would be elegant if we can prove that in the step-wise interpolation method of Section 3.3, we can always choose the optimal path; see Section 3.3.1.

- Study the performance of the module minimisation algorithms in the Guruswami–Sudan where we use different multiplicities for different points. This variant of Guruswami–Sudan is used in soft-decision decoding [KV03a] or decoding of Alternant codes, see Section 3.6 and Section 5.3.2. Lee and O'Sullivan gave a simple algorithm for producing a basis [LO06], but did not write the basis explicitly, which makes analysis on the orthogonality defect more difficult; it does not seem difficult, however, to extend the arguments of Theorem 3.24 on page 60 to encompass varying multiplicities. As we have seen, the module minimisation of Lee and O'Sullivan is not the fastest, which means that pointing out that one should use the Alekhnovich or the GJV algorithm will lead to improved complexity.
- There are several things which needs to be investigated in our Power Gao decoding of Hermitian codes. Most prominently, the relation of Power Gao to Power Syndromes, in particular extending Proposition 4.10 on page 103; extending Proposition 4.7 on page 101; and how to incorporate majority voting, see discussion in Section 4.5.
- The interpolation algorithm for Hermitian codes of Section 3.5 should be readily generalisable to a larger class of AG codes; in particular, the simple C_{ab} curves, also handled by Brander in [Bra10]. The same goes for the Power Gao decoder for Hermitian codes.

Improving the results

These are slightly larger projects which would complement and improve understanding of this thesis' results.

- The asymptotic analyses we have performed for the range of different decoding approaches reveal an interesting fact: they are basically all equally fast! This is of course in an asymptotic setting, and is also disregarding certain important details such as the Wu decoder using different parameters than the Guruswami–Sudan, but it is still an important observation. To evaluate which method will perform best for concrete parameter choices, we need to improve our analysis, e.g. by re-analysing the methods using asymptotic expansion (i.e. retaining the constant factor of the leading term of the asymptotic expression throughout).
- We have in this thesis exclusively focused on the worst-case running time. Of greater import, at least on software platforms, is the *average* running time, and it would therefore be interesting to examine how the module minimisation algorithms perform in this metric. In particular, informal (and incomplete) experiments I have performed with Codinglib [Nie13a] seem to indicate that the number of row reductions performed by Mulders–Storjohann is usually close to the bound, but that the degrees of the matrices multiplied together by the Alekhnovich algorithm when "merging" its computation trees are of much lower degree than expected. These merges dominate the complexity, and my conjecture is that the average running time of the Alekhnovich algorithm is therefore much

better. Of course, it is necessary to specify, in one way or another, from which distribution the input comes, and this might be difficult to do in a formal way with the matrices coming from decoding instances.

- This thesis opens up two new approaches for upper bounding the failure probability of Power decoding: by examining Power Gao decoding, or by examining Coppersmith–Sudan. Either of these seem to be more amenable to analysis than the original Power Syndromes, so it is my hope that such an analysis would be successful.

- We saw how to Wu list decode binary Goppa codes up to the binary Johnson bound, and the obvious question is if it is possible to decode q-ary Goppa codes, or even q-ary Alternant codes, up to the q-ary Johnson bound with the Wu paradigm. In the Goppa case, the obvious idea is to do a q-ary decomposition of Λ, similar to the decomposition in $a(x)$ and $b(x)$ that we did; however, it is not clear how this relates to the error evaluator, and how to get key equations which could be Wu list decoded afterwards.

- It would be very interesting to understand Sakata's algorithm [Sak90] in light of module minimisation. This algorithm works over N dimensions, so forcing the problem down into a matrix problem over $\mathbb{F}[x]$ might imply explosively large matrices; on the other hand, Sakata's algorithm itself needs a large number of "auxiliary polynomials", which would likely correspond to rows of this big matrix. The question probably boils down to whether the seemingly dynamic and lazy-evaluated nature of Sakata's algorithm can be captured by a statically sized $\mathbb{F}[x]$ matrix. In the case of classical key equation decoding Hermitian codes, i.e. the $\ell = 1$ in Section 4.4, module minimisation was at least as fast as Sakata's algorithm, which sets $N = 2$ in this case. In case such a thing is tractable in general, it is possible that the notion of 2D key equation is not powerful enough to encompass what Sakata's algorithm can handle so we might need a generalisation. One should then also look at generalising the Demand–Driven algorithm, and this generalisation might even lead to an algorithm closely related to Sakata's.

- Kötter proposed a parallel variant of the Berlekamp–Massey extension to AG codes in [Köt98], providing a significant speedup in hardware or on multi-processor software architectures. It would be interesting to investigate whether a similar parallelism is possible in Mulders–Storjohann or the Demand–Driven algorithm; especially the latter is very close to the Berlekamp–Massey algorithm, so this might well be possible. The D&C variants are less amenable, since they seem to need the entire problem when "merging" computation trees using fast multiplication techniques.

- As discussed in Section 3.6, Lee and O'Sullivan's approach for finding Q for Guruswami–Sudan decoding of Hermitian codes has been generalised to a larger class of AG codes [GMR12]. It would be interesting to see whether the faster D&C module minimisation techniques apply to in this work as well.

Extending the results

These are ideas for more vague extensions and generalisations; I am not sure how precisely they should be done or if they are even possible.

- The Wu paradigm can be seen as a way to solve a simple key equation, i.e. $\sigma = \rho = 1$, when one has certain knowledge on the zeroes of the sought polynomials. If it was possible to extend this for $\sigma > 1$, then that would have immediate applications in extending Power decoding (performing this Multi-Wu after failed Power decoding), and in other areas where Multi-LFSR solving has been useful, e.g. Interleaved Reed–Solomon codes and decoding cyclic codes. Something like this also seems necessary to formulate a Wu list decoder for AG codes.

  The first step holds completely analogous when $\sigma > 1$, i.e. by Proposition 2.14 on page 17 the sought solution is a small $\mathbb{F}[x]$-linear combination of the found weak Popov form. The next step is more difficult, that is, we should perform a kind of multi-dimensional interpolation through at least τ of n points from $\mathbb{F} \times \mathbb{P}^{\ell-1}(\mathbb{F})$. The generalisation from Guruswami–Sudan to rational interpolation is a small one, since it just moves from $\mathbb{F} \times \mathbb{F}$ to $\mathbb{F} \times \mathbb{P}^1(\mathbb{F})$, and we are therefore here talking about a conceptually broader generalisation.
- As we have discussed, Power decoding has strong ties to Sudan decoding, that is Guruswami–Sudan when $s = 1$. It is quite obvious to seek a Power decoding with multiplicities, i.e. $s > 1$, and many people have tried. Inspired by the Q-finding approaches of Chapter 3, it seems clear that such a generalisation should incorporate $G(x)$ to higher powers; however, I have failed in finding a key equation relation for such higher G, and perhaps the necessary relation will not take this form. A possibly fruitful approach is to study the Power decoding $\leftrightarrow$ Coppersmith–Sudan relation of Section 4.3.2: the latter algorithm already supports $s > 1$, and it might be possible to generalise the relation, thereby arriving by a back door at a Power decoding with multiplicities.
- There is a number of clear connections between the Guruswami–Sudan list decoder for GRS codes and the Wu decoder; notably of course the decoding radius, but also the duality in parameter choice given in Section 5.2.2. It feels like there should be a closer algebraic connection which could more satisfactorily explain these agreements, just as we explained a closer algebraic connection between Power decoding and Sudan decoding in Section 4.3.
- It would be very interesting to generalise the multi-trial decoding algorithm of Section 3.3 for the soft-decision Kötter–Vardy algorithm with varying multiplicities. If this was possible, the result could very well be a soft-decision list decoder, with fast enough average case complexity for practical use on e.g. software platforms.

Appendix A

List of symbols and notation

This is a short description of most of the "globally defined" symbols and operators used throughout the thesis; in particular, they contain the ones that we might use far from their original definition.

Note that in a few sections, some of the symbols are redefined and used in an analogous but different manner; for instance, many code-pertaining symbols are redefined for Hermitian codes in Section 4.4. Only the common definition is given here.

$\mathbb{F}$	Some given finite field.
$\mathcal{C}$	The current code in question, usually a GRS code.
n	The length of the codewords in $\mathcal{C}$.
k	The dimension of $\mathcal{C}$. For GRS codes, all codewords are obtained by evaluating (with $\operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$) a polynomial of degree at most $k-1$.
d	The minimum distance of the code. For GRS codes $d = n - k + 1$.
$\operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$	The evaluation function for producing codewords in the GRS code $\mathcal{C}$: for $f \in \mathbb{F}[x]$ then $\operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f) = \big(\beta_1 f(\alpha_1), \dots, \beta_n f(\alpha_n)\big)$.
$\boldsymbol{c}$	$= (c_1, \dots, c_n)$. The sent codeword.
$\boldsymbol{r}$	$= (r_1, \dots, r_n) = \boldsymbol{c} + \boldsymbol{e}$. The received codeword.
$f(x)$	The evaluated information word; i.e. $\boldsymbol{c} = \operatorname{ev}_{\boldsymbol{\alpha},\boldsymbol{\beta}}(f)$.
$\boldsymbol{\alpha}$	The evaluation points for the GRS code $\mathcal{C}$. n distinct elements of $\mathbb{F}$.
$\boldsymbol{\beta}$	The column multipliers for the GRS code $\mathcal{C}$. n non-zero elements of $\mathbb{F}$.
$\boldsymbol{r}'$	$= (r_1/\beta_1, \dots, r_n/\beta_n)$. Short-hand.
$\boldsymbol{e}$	$= (e_1, \dots, e_n)$. The error vector.
$\mathcal{E}$	$= \{i \mid e_i \neq 0\}$. The error positions.
τ	Usually a decoding radius. Also as a function $\tau(s, \ell)$ meaning the greatest integer such that $E_{\mathrm{GS}}^{[n,k]}(s, \ell, \tau) > 0$. In Section 5.1, the number of points through which a solution to the rational interpolation problem should at least pass.
s	The "multiplicity" when performing Guruswami–Sudan decoding or Wu decoding.
ℓ	The "list size" when performing Guruswami–Sudan decoding or Wu decoding.
$E_{\mathrm{GS}}^{[n,k]}$	Guruswami–Sudan satisfiability function for the parameters; a Q polynomial exists for the parameters if $E_{\mathrm{GS}}^{[n,k]}(s, \ell, \tau) > 0$ for the given code $\mathcal{C}$. Definition 3.3 on page 49.
$E_{\mathrm{Wu}}^{[n,\theta]}$	Wu satisfiability function, similar to $E_{\mathrm{GS}}^{[n,k]}$, but dependent on the rational interpolation problem. For decoding a GRS code, $\theta = 2\tau - d$. Definition 3.3 on page 49.
$J(n, d)$	$= n - \sqrt{n(n-d)}$. The asymptotic Johnson radius. An upper bound on the decoding radius of both the Guruswami–Sudan and Wu list decoders. Section 3.6.
$J_q(n, d)$	$= \frac{q-1}{q}\left(n - \sqrt{n\left(n - \frac{q}{q-1}d\right)}\right)$. The q-ary Johnson radius. The decoding radius of the Wu list decoder for binary Goppa codes, where $d = 2 \deg G + 1$. Section 3.6 and Section 5.4.

$G(x)$ $= \prod_{i=1}^{n}(x-\alpha_i)$. Definition 3.22 on page 60.

$R(x)$ The Lagrangian through $\{(\alpha_i, r'_i)\}$. Definition 3.22 on page 60.

$R^{(t)}(x)$ The Lagrangian through $\{(\alpha_i, r'^t_i)\}$. (4.1) on page 95.

$\Lambda(x)$ $= \prod_{i\in\mathcal{E}}(x-\alpha_i)$. The error locator. Definition 4.1 on page 95.

$\Omega^{(t)}(x)$ $= \sum_{j\in\mathcal{E}} \zeta_j e_j^{(t)} \frac{\Lambda(x)}{x-\alpha_j}$. The powered error evaluator. Definition 4.1 on page 95.

ζ_i $= \prod_{j\neq i}(\alpha_i-\alpha_j)^{-1}$. See page 95.

$S_\circ^{(a,b)}(x)$ $= \overline{R}(x)^{a_R}/\overline{G}(x)^{a_G}$. Power series related to "powered" syndromes. (3.9) on page 72.

$S_\bullet^{(a)}(x)$ $= \overline{R}^{(a)}(x)/\overline{G}(x)$. "Powered" syndromes power series. (4.4) on page 102, but see also Proposition 4.11 on page 104.

∇_t^Q An upper bound on the x-degree of $Q_{[t]}(x)$ for polynomial or rational interpolation. $\nabla_t^Q = s(n-\tau) - t(k-1)$ for Guruswami–Sudan, while $\nabla_t^Q = s\tau - t\theta_1 - (\ell - t)\theta_2$ for rational interpolation.

Operators

$\deg \boldsymbol{v}$	If $v \in \mathbb{F}[x]^m$ then $\deg \boldsymbol{v} = \max_i\{\deg v_i\}$.
$\deg V$	If $V \in F[x]^{m\times n}$ then $\deg V = \sum_i \deg \boldsymbol{v}_i$, i.e. it's "summed row-degree".
$\operatorname{maxdeg} V$	If $V \in \mathbb{F}[x]^{m\times n}$ then $\operatorname{maxdeg} V = \max_{i,j}\{\deg v_{i,j}\}$.
$\operatorname{LP}(\boldsymbol{v})$	$= \max\{j \mid \deg v_j = \deg \boldsymbol{v}\}$, $\boldsymbol{v} \in \mathbb{F}[x]^m$. Note that if more than one entry in $\boldsymbol{v}$ has degree $\deg \boldsymbol{v}$, it is the *right-most* entry which is the leading position.
$\operatorname{LP}_{\preceq}(\boldsymbol{v})$	The index i such that $v_i\boldsymbol{e}_i \succeq v_j\boldsymbol{e}_j$ for $j \neq i$, where $\boldsymbol{v} \in \mathbb{F}[x]^m$ and $\preceq$ is a module monomial ordering. Section 2.1.2.
$\operatorname{LT}(\boldsymbol{v})$	$= v_{\operatorname{LP}(\boldsymbol{v})}$, $\boldsymbol{v} \in \mathbb{F}[x]^m$. Also $\operatorname{LT}_{\preceq}(\boldsymbol{v}) = v_{\operatorname{LP}_{\preceq}(\boldsymbol{v})}$.
$\psi(\boldsymbol{v})$	$= m\deg \boldsymbol{v} + \operatorname{LP}(\boldsymbol{v})$, where $\boldsymbol{v} \in \mathbb{F}[x]^m$. Definition 2.20 on page 19.
$\deg_{(w_1,\ldots,w_\kappa)}$	If $p = \alpha\prod x_i^{\theta_i} \in \mathbb{F}[x_1,\ldots,x_\kappa]$ then $\deg_{(w_1,\ldots,w_\kappa)} p = \sum_{i=1}^{\kappa} w_i\theta_{j,i}$. If p is a sum of such monomials with distinct exponent lists, then $\deg_{(w_1,\ldots,w_\kappa)} p$ is the maximal of each monomial's weighted degree.
$\deg p$	If $p(x) \in \mathbb{F}[x]$, then just the usual degree. If $p \in \mathbb{F}[x_1,\ldots,x_\kappa]$, then $\deg p = \deg_{(1,\ldots,1)} p$.
$\deg_{x_i} p$	$= \deg_{(0,\ldots,0,1,0,\ldots,0)}$ where the 1 is on the ith position.
$\operatorname{LC}(p)$	If $p(x) = \sum_i p_i x^i \in \mathbb{F}[x]$ then $\operatorname{LC}(p) = p_{\deg p}$.
$Q_{[t]}$	If $Q = \sum_i Q_i(x)y^i \in \mathbb{F}[x,y]$ then $Q_{[t]} = Q_i(x)$. If $Q = \sum_i Q_i(x)y^i z^{\ell-i} \in \mathbb{F}[x][y:z]^\ell$ then $Q_{[t]} = Q_i$. See also below for its meaning in the Hermitian codes sections.
$\Phi_{\nu,\boldsymbol{w}}$	The weight-embedding function, Definition 2.11 on page 16: $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}) = \left(x^{w_1}v_1(x^\nu),\ldots,x^{w_m}v_m(x^\nu)\right)$
$\preceq_{\nu,\boldsymbol{w}}$	Weighted module monomial ordering. Definition 2.9 on page 15.
$\trianglelefteq$	Used to depict degree upper bounds on vectors and matrices containing polynomials. If $p \in \mathbb{F}[x]$ then $p \trianglelefteq n$ for any $n \geq \deg p$. $\trianglelefteq$ is then extended element-wise to vectors and matrices.
${}^{[d]}\overline{p}$	For $p \in \mathbb{F}[x]$ with $\deg p \leq d$ then ${}^{[d]}\overline{p} = x^d p(x^{-1})$. We often simply write $\overline{p}$ when d as an upper bound on the degree of p is implied.
$\operatorname{pos}(x)$	$= x$ for $x > 0$ and $\operatorname{pos}(x) = 0$ for $x \leq 0$.

Complexity analysis

$\mathrm{P}(t)$	Complexity of multiplying together two polynomials of degree up to t. We relax it as $\mathrm{P}(t) = O(t\log t\log\log t)$.
$\mathrm{M}(m)$	Complexity of multiplying together two matrices of $\mathbb{F}^{m\times m}$. We relax it as the trivial $\mathrm{M}(m) = O(m^3)$. Note that one can multiply together two matrices of $\mathbb{F}[x]^{m\times m}$, each of max-degree at most t, in time $\mathrm{M}(m)\mathrm{P}(t)$.

See also the description in Section 1.2.

Hermitian codes

q	The Hermitian code is defined over $\mathbb{F}_{q^2}$ and has length up to q^3.
$\mathcal{H}(X,Y)$	$= Y^q + Y - X^{q+1}$; the Hermitian curve equation.
F	$= \mathbb{F}_{q^2}(x,y)$ where x, y satisfy $\mathcal{H}(x,y) = 0$. The Hermitian function field.
g	$= \frac{1}{2}q(q-1)$; the genus of the Hermitian curve.
P_i	The rational points of the Hermitian curve; for $i = \infty$, the point at infinity.
$P_{\alpha,j}$	The rational points grouped together. $\alpha \in \mathbb{F}_{q^2}$, and $P_{\alpha,j}$ are the points lying above the zero of $x - \alpha$ for $j = 1, \ldots, q$. Thus $(x-\alpha) = \mathcal{L}(-\sum_{j=1}^{q} P_{\alpha,j} + qP_\infty)$; Proposition 3.44 on page 79.
$\mathcal{P}$	$= \{P_1, \ldots, P_n, P_\infty\}$.
$\mathcal{P}^\star$	$= \{P_1, \ldots, P_n\}$.
$\mathcal{R}$	$= \mathbb{F}_{q^2}[x,y] = \mathcal{L}(\infty P_\infty)$.
m	Maximal pole order at P_∞ of functions evaluated to construct the Hermitian code; Definition 3.48 on page 80.
G	$= \prod_{i=1}^{n/q}(x - \alpha_i) \in \mathcal{R}$; Definition 3.54 on page 83.
R	Non-zero element of $\mathcal{R}$ satisfying $R(P_i) = r_i$ for $i = 1, \ldots, n$; Definition 3.54 on page 83.
$R^{(t)}$	Non-zero elements of $\mathcal{R}$ satisfying $R^{(t)}(P_i) = r_i^t$ for $i = 1, \ldots, n$ and $t \in \mathbb{N}_0$; (4.7) on page 113.
Λ	The error locator: the non-zero polynomial in $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$ with minimal $\deg_{\mathcal{H}}$ and $\mathrm{LC}_{\mathcal{H}}(\Lambda) = 1$; Definition 4.18 on page 113
$\deg_{\mathcal{H}}$	$: \mathcal{R} \mapsto \mathbb{N}_0 \cup \{-\infty\}$ by $\deg_{\mathcal{H}} p = -v_{P_\infty}(p)$ or equivalently $\deg_{\mathcal{H}}(x^i y^j) = qi + (q-1)j$ if $j < q$; Definition 3.45 on page 79.
$\deg_{\mathcal{H},w}$	If $Q \in \mathcal{R}[z]$ with $Q(z) = \sum_{t=0}^{\deg_z Q} Q_t(x,y)z^t$, then $\deg_{\mathcal{H},w} Q = \max\{\deg_{\mathcal{H}} Q_t + tw\}$.
$Q_{[t]}$	If $Q \in \mathcal{R}[z]$ with $Q(z) = \sum_{t=0}^{\deg_z Q} Q_t(x,y)z^t$, then $Q_{[t]} = Q_t$.
$\Upsilon(\cdot)$	$: \mathcal{R} \mapsto \mathbb{F}_{q^2}[x]^q$ by $\Upsilon(\sum_{i=0}^{q-1} y^i g_i(x)) = \big(g_0, \ldots, g_{q-1}\big)$; page 84.
$\Upsilon_z(\cdot)$	$: \mathcal{R}[z]_\ell \mapsto \mathbb{F}_{q^2}[x]^{(\ell+1)q}$, where $\mathcal{R}[z]_\ell$ are elements of $\mathcal{R}[z]$ of degree at most ℓ, by $\Upsilon_z(\sum_{t=0}^{\ell} Q_t z^t) = \big(\Upsilon(Q_0) \mid \ldots \mid \Upsilon(Q_\ell)\big)$; page 84.
$\amalg(\cdot)$	$: \mathcal{R} \mapsto \mathbb{F}_{q^2}^{q\times(2q-2)}$ is the multiplication matrix of (3.14) on page 84.
Ξ	The $\mathcal{H}$ reduction matrix of (3.15) on page 84.

Bibliography

[ABC10] D. Augot, M. Barbier, and A. Couvreur. List-decoding of binary Goppa codes up to the binary Johnson bound. *arXiv*, abs/1012.3439, 2010.

[Ajt98] M. Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions. In *ACM symposium on Theory of Computing*, pages 10–19. ACM, 1998.

[AK11] M. Ali and M. Kuijper. A Parametric Approach to List Decoding of Reed-Solomon Codes Using Interpolation. *IEEE Transactions on Information Theory*, 57(10):6718–6728, 2011.

[Ale05] M. Alekhnovich. Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed–Solomon Codes. *IEEE Transactions on Information Theory*, 51(7), 2005.

[BB10] P. Beelen and K. Brander. Key equations for list decoding of Reed–Solomon codes and how to solve them. *Journal of Symbolic Computation*, 45(7):773–786, 2010.

[Bee13] P. Beelen. Personal communication, 2013.

[Ber68] E. R. Berlekamp. *Algebraic Coding Theory*. Aegean Park Press, 1968.

[Ber96] E. R. Berlekamp. Bounded Distance +1 Soft-Decision Reed-Solomon Decoding. *IEEE Transactions on Information Theory*, 42(3):704–720, 1996.

[Ber11a] D. J. Bernstein. Simplified high-speed high-distance list decoding for alternant codes, 2011.

[Ber11b] D. J. Bernstein. List Decoding for Binary Goppa Codes. In *International Workshop on Coding Theory and Cryptography*, pages 62–80, 2011.

[BGM96] G. Baker and P. Graves-Morris. *Padé approximants*, volume 59. Cambridge University Press, 1996.

[BH08a] P. Beelen and T. Høholdt. The Decoding of Algebraic Geometry Codes. In E. Martínez-Moro, editor, *Advances in algebraic geometry codes*, volume 5. World Scientific Publishing Company, 2008.

[BH08b] P. Beelen and T. Høholdt. List decoding using syndromes. In J. Chaumine, J. Hirschfeld, and R. Rolland, editors, *Algebraic Geometry and its Applications, vol. 5, Series on Number Theory and its Applications*, pages 315–331. World Scientific, 2008.

[BHNW13] P. Beelen, T. Høholdt, J. S. R. Nielsen, and Y. Wu. On Rational-Interpolation Based List-Decoding and List-Decoding Binary Goppa Codes. *IEEE Transactions of Information Theory*, 59(6):3269–3281, 2013.

[BK12] I. I. Bouw and S. Kampf. Syndrome decoding for hermite codes with a sugiyama-type algorithm. *Advances in Mathematics of Communications*, 6(4):419–442, 2012.

[BL92] B. Beckermann and G. Labahn. A uniform approach for Hermite Padé and simultaneous Padé approximants and their matrix-type generalizations. *Numerical Algorithms*, 3(1):45–54, 1992.

[BL94] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, 1994.

[BL09] B. Beckermann and G. Labahn. Fraction-Free Computation of Simultaneous Padé Approximants. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pages 15–22. ACM, 2009.

[Bos99] M. Bossert. *Channel Coding for Telecommunications*. John Wiley & Sons, Inc., 1999.

[Bra10] K. Brander. *Interpolation and List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2010.

[CB04] Y. Cassuto and J. Bruck. A Combinatorial Bound on the List Size. Technical report, California Institute of Technology, 2004.

[CBM13] Y. Cassuto, J. Bruck, and R. J. McEliece. On the Average Complexity of Reed–Solomon List Decoders. *IEEE Transactions on Information Theory*, 59(4):2336–2351, 2013.

[CH10] H. Cohn and N. Heninger. Ideal forms of Coppersmith's theorem and Guruswami–Sudan list decoding. *preprint*, arXiv:1008.1284, 2010.

[CHGN08] D. Coppersmith, N. Howgrave-Graham, and S. Nagaraj. Divisors in residue classes, constructively. *Mathematics of Computation*, 77(261):531–545, 2008.

[CLO98] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*, volume 185. Springer Verlag, 1998.

[Coo66] S. A. Cook. *On the minimum computation time of functions*. PhD thesis, Department of Mathematics, Harvard University, 1966.

[Cop97] D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[CS03] D. Coppersmith and M. Sudan. Reconstructing Curves in Three (and Higher) Dimensional Space from Noisy Data. In *ACM Symposium on Theory of Computing*, pages 136–142. ACM, 2003.

[Dor87] J. Dornstetter. On the Equivalence Between Berlekamp's and Euclid's Algorithms. *IEEE Transactions on Information Theory*, 33(3):428–431, 1987.

[Eli57] P. Elias. List decoding for noisy channels. Technical report, Research Laboratory of Electronics, MIT, 1957.

[Fit95] P. Fitzpatrick. On the Key Equation. *IEEE Transactions on Information Theory*, 41(5):1290–1302, 1995.

[FT89] G.-L. Feng and K. K. Tzeng. A Generalized Euclidean Algorithm for Multisequence Shift-Register Synthesis. *IEEE Transactions on Information Theory*, 35(3):584–594, 1989.

[FT91] G.-L. Feng and K. K. Tzeng. A Generalization of the Berlekamp-Massey Algorithm for Multisequence Shift-Register Synthesis with Applications to Decoding Cyclic Codes. *IEEE Transactions on Information Theory*, 37(5):1274–1287, 1991.

[Für09] M. Fürer. Faster Integer Multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009.

[FWRT94] G.-L. Feng, V. K. Wei, T. R. N. Rao, and K. K. Tzeng. Simplified Understanding and Efficient decoding of a Class of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 40(4):981–1002, 1994.

[Gao03] S. Gao. A new algorithm for decoding reed-solomon codes. *Kluwer International Series in Engineering and Computer Science*, pages 55–68, 2003.

[GJV03] P. Giorgi, C. Jeannerod, and G. Villard. On the Complexity of Polynomial Matrix Computations. In *Proceedings of International Symposium on Symbolic and Algebraic Computation '03*, pages 135–142. ACM, 2003.

[GMR12] O. Geil, R. Matsumoto, and D. Ruano. List Decoding Algorithms Based on Gröbner Bases for General One-Point AG Codes. In *IEEE*

International Symposium on Information Theory, pages 86–90. IEEE, 2012.

[Gop70] V. D. Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.

[GR06] V. Guruswami and A. Rudra. Explicit Capacity-Achieving List-Decodable Codes. In *Proceedings of STOC '06*, pages 1–10. ACM, 2006.

[GRS00] O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries: the highly noisy case. *SIAM Journal on Discrete Mathematics*, 13:535–570, 2000.

[GS99] V. Guruswami and M. Sudan. Improved Decoding of Reed–Solomon Codes and Algebraic Geometry Codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[GSS00] V. Guruswami, A. Sahai, and M. Sudan. "Soft-decision" Decoding of Chinese Remainder Codes. In *International Symposium on Foundations of Computer Science*, pages 159–168. IEEE, 2000.

[Gur01] V. Guruswami. *List decoding of error-correcting codes*. PhD thesis, Massachusetts Institute of Technology, 2001.

[Gur07] V. Guruswami. Algorithmic Results in List Decoding. *Foundations and Trends in Theoretical Computer Science*, 2(2):107–195, 2007.

[GW12] V. Guruswami and C. Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Transactions of Information Theory*, 59:3257–3268, 2012.

[GX12] V. Guruswami and C. Xing. Folded codes from function field towers and improved optimal rate list decoding. In *ACM Symposium on Theory of Computing*, pages 339–350. ACM, 2012.

[Han01] J. P. Hansen. Dependent rational points on curves over finite fields-Lefschetz theorems and exponential sums. In *Workshop on Coding and Crypthography*, pages 297–309, 2001.

[Her78] C. Hermite. Sur la Formule d'Interpolation de Lagrange. *Journal für die Reine und Angewandte Mathematik*, 84(1):70–79, 1878.

[HG01] N. Howgrave-Graham. Approximate integer common divisors. In *Cryptography and Lattices*, pages 51–66. Springer, 2001.

[HJ00] A. E. Heydtmann and J. M. Jensen. On the Equivalence of the Berlekamp-Massey and the Euclidean Algorithms for Decoding. *IEEE Transactions on Information Theory*, 46(7):2614–2624, 2000.

[JH01] J. Justesen and T. Høholdt. Bounds on List Decoding of MDS Codes. *IEEE Transactions on Information Theory*, 47(4):1604–1609, 2001.

[JH04] J. Justesen and T. Høholdt. *A Course in Error-Correcting Codes*. European Mathematical Society, 2004.

[JLJ+89] J. Justesen, K. J. Larsen, H. E. Jensen, A. Havemose, and T. Høholdt. Construction and Decoding of a Class of Algebraic Geometry Codes. *IEEE Transactions on Information Theory*, 35(4):811–821, 1989.

[JLJH92] J. Justesen, K. J. Larsen, H. E. Jensen, and T. Høholdt. Fast Decoding of Codes From Algebraic Plane Curves. *IEEE Transactions on Information Theory*, 38(1):111–119, 1992.

[JNH99] H. E. Jensen, R. R. Nielsen, and T. Høholdt. Performance Analysis of a Decoding Algorithm for Algebraic-Geometry Codes. *IEEE Transactions on Information Theory*, 45(5):1712–1717, 1999.

[Joh62] S. M. Johnson. A New Upper Bound for Error-Correcting Codes. *IEEE Transactions on Information Theory*, 46:203–207, 1962.

[Kam11] S. Kampf. *Decoding Hermitian Codes – An Engineering Approach*. PhD thesis, Universität Ulm, 2011.

[KL13] S. Kampf and W. Li. Decoding Interleaved Reed–Solomon and Hermitian Codes with Generalized Divisions. In *ITG Conference on Systems, Communications and Coding*, 2013.

[KMV11] R. Kötter, J. Ma, and A. Vardy. The Re-Encoding Transformation in Algebraic List-Decoding of Reed–Solomon Codes. *IEEE Transactions on Information Theory*, 57(2):633–647, February 2011.

[KO64] A. Karatsuba and Y. Ofman. Multiplication of Many-Digital Numbers by Automatic Computers. *Proceedings of the USSR Academy of Sciences*, 145:293–294, 1964.

[Köt96] R. Kötter. *On Algebraic Decoding of Algebraic-Geometric and Cyclic Codes*. PhD thesis, University of Linköping, 1996.

[Köt98] R. Kötter. A Fast Parallel Implementation of a Berlekamp-Massey Algorithm for Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 44(4):1353–1368, 1998.

[KV03a] R. Kötter and A. Vardy. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003.

[KV03b] R. Kötter and A. Vardy. A Complexity Reducing Transformation in Algebraic List Decoding of Reed–Solomon Codes. In *Information Theory Workshop, 2003. Proceedings. 2003 IEEE*, pages 10–13. IEEE, 2003.

[Lau03] N. Lauritzen. *Concrete abstract algebra: from numbers to Gröbner bases*. Cambridge University Press, 2003.

[LBAO12] K. Lee, M. Bras-Amorós, and M. E. O'Sullivan. Unique Decoding of Plane AG Codes via Interpolation. *IEEE Transactions of Information Theory*, 58(6):3941–3950, 2012.

[Len85] A. Lenstra. Factoring Multivariate Polynomials over Finite Fields. *Journal of Computer and System Sciences*, 30(2):235–248, 1985.

[Leo06] V. K. Leontév. Roots of Random Polynomials over a Finite Field. *Mathematical Notes*, 80(1):300–304, 2006.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LO06] K. Lee and M. E. O'Sullivan. An Interpolation Algorithm Using Gröbner Bases for Soft-Decision Decoding of Reed-Solomon Codes. In *Information Theory, 2006 IEEE International Symposium on*, pages 2032–2036. IEEE, 2006.

[LO08] K. Lee and M. E. O'Sullivan. List Decoding of Reed–Solomon Codes from a Gröbner Basis Perspective. *Journal of Symbolic Computation*, 43(9):645 – 658, 2008.

[LO09] K. Lee and M. E. O'Sullivan. List decoding of Hermitian codes using Gröbner bases. *Journal of Symbolic Computation*, 44(12):1662–1675, 2009.

[LO10] K. Lee and M. E. O'Sullivan. Algebraic Soft-Decision Decoding of Hermitian Codes. *IEEE Transactions on Information Theory*, 56(6):2587–2600, 2010.

[LSN13] W. Li, V. Sidorenko, and J. S. R. Nielsen. On Decoding Interleaved Chinese Remainder Codes. In *IEEE International Symposium on Information Theory*, 2013.

[Mas69] J. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

[May03] A. May. *New RSA vulnerabilities using lattice reduction methods*. PhD thesis, University of Paderborn, 2003.

[McE03] R. McEliece. The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes. *IPN progress report*, pages 42–153, 2003.

[MS77] F. J. MacWilliams and N. J. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Publishing, 1977.

[MS03] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35(4):377–401, 2003.

[NH98] R. R. Nielsen and T. Høholdt. Decoding Reed–Solomon codes beyond half the minimum distance. In *Coding Theory, Cryptography and Related Areas*, pages 221–236. Springer-Verlag, 1998.

[Nie10] J. S. R. Nielsen. List-decoding of error-correcting codes. Master's thesis, DTU, Technical University of Denmark, 2010.

[Nie13a] J. S. R. Nielsen. Codinglib: An Algebraic Coding Theory Library for Sage, 2013. http://jsrn.dk/codinglib.

[Nie13b] J. S. R. Nielsen. Generalised Multi-sequence Shift-Register Synthesis using Module Minimisation. In *IEEE International Symposium on Information Theory*, 2013.

[NZ13] J. S. R. Nielsen and A. Zeh. Multi-Trial Guruswami–Sudan Decoding for Generalised Reed–Solomon Codes. In *Workshop on Coding and Cryptography*, 2013.

[OBA08] M. E. O'Sullivan and M. Bras-Amorós. The Key Equation for One-Point Codes. In E. Martínez-Moro, editor, *Advances in algebraic geometry codes*, volume 5. World Scientific Publishing Company, 2008.

[O'K03] H. O'Keeffe. *Gröbner Basis techniques for certain problems in coding and systems theory*. PhD thesis, University College Cork, 2003.

[OS06] Z. Olesh and A. Storjohann. The vector rational function reconstruction problem, 2006.

[Pat75] N. Patterson. The Algebraic Decoding of Goppa Codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.

[Pop70] V. Popov. Some properties of the control systems with irreducible matrix-transfer functions. In *Seminar on Differential Equations and Dynamical Systems, II*, pages 169–180. Springer, 1970.

[Rot06] R. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[RR00] R. Roth and G. Ruckenstein. Efficient Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance. *IEEE Transactions on Information Theory*, 46(1):246 –257, 2000.

[Ruc01] G. Ruckenstein. *Error decoding strategies for algebraic codes*. PhD thesis, Israel Institute of Technology, Haifa, 2001.

[S$^+$] W. A. Stein et al. *Sage Mathematics Software*. The Sage Development Team. http://www.sagemath.org.

[Sak90] S. Sakata. Extension of the Berlekamp–Massey algorithm to N dimensions. *Information and Computation*, 84(2):207–239, 1990.

[SB11] V. Sidorenko and M. Bossert. Fast skew-feedback shift-register synthesis. *Designs, Codes and Cryptography*, pages 1–13, 2011.

[SJH95] S. Sakata, H. E. Jensen, and T. Høholdt. Generalized Berlekamp-Massey Decoding of Algebraic-Geometric Codes up to Half the Feng–Rao Bound. *IEEE Transactions on Information Theory*, 41(6):1762–1768, 1995.

[SKHN75] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A Method for Solving Key Equation for Decoding Goppa Codes. *Information and Control*, 27(1):87–99, 1975.

[SKHN76] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. Further Results on Goppa Codes and Their Applications to Constructing Efficient Binary Codes. *IEEE Transactions on Information Theory*, 22(5):518–526, 1976.

[SS06] G. Schmidt and V. Sidorenko. Multi-Sequence Linear Shift-Register Synthesis: The Varying Length Case. In *Information Theory, 2006 IEEE International Symposium on*, pages 1738–1742. IEEE, 2006.

[SS11] V. Sidorenko and G. Schmidt. A Linear Algebraic Approach to Multisequence Shift-Register Synthesis. *Problems of Information Transmission*, 47(2):149–165, 2011.

[SSB06] G. Schmidt, V. Sidorenko, and M. Bossert. Decoding Reed-Solomon codes beyond half the minimum distance using shift-register synthesis. In *Information Theory, 2006 IEEE International Symposium on*, pages 459–463. IEEE, 2006.

[SSB09] G. Schmidt, V. Sidorenko, and M. Bossert. Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs. *Information Theory, IEEE Transactions on*, 55(7):2991–3012, 2009.

[SSB10] G. Schmidt, V. Sidorenko, and M. Bossert. Syndrome Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis. *IEEE Transactions on Information Theory*, 56(10):5245–5252, 2010.

[Sti88] H. Stichtenoth. A Note on Hermitian Codes Over $GF(q)$. *IEEE Transactions on Information Theory*, 34(5):1345–1348, 1988.

[Sti09] H. Stichtenoth. *Algebraic Function Fields and Codes*. Springer, 2nd edition, 2009.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[Sud97] M. Sudan. Decoding of Reed–Solomon Codes beyond the Error-Correction Bound. *Journal of Complexity*, 13(1):180–193, 1997.

[SW98] M. A. Shokrollahi and H. Wasserman. Decoding algebraic-geometric codes beyond the error-correction bound. In *ACM Symposium on Theory of Computing*, pages 241–248. ACM, 1998.

[SW99] M. A. Shokrollahi and H. Wasserman. List Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 45(2):432–437, 1999.

[TCM12] S. Tang, L. Chen, and X. Ma. Progressive List-Enlarged Algebraic Soft Decoding of Reed-Solomon Codes. *IEEE Communications Letters*, 16(6):901–904, June 2012.

[TL12] P. Trifonov and M. Lee. Efficient Interpolation in the Wu List Decoding Algorithm. *IEEE Transactions on Information Theory*, 58(9):5963–5971, 2012.

[Tri10a] P. V. Trifonov. Another Derivation of Wu List Decoding Algorithm and Interpolation in Rational Curve Fitting. In *Proc. of IEEE R8 SIBIRCON*, pages 59–64, 2010.

[Tri10b] P. V. Trifonov. Efficient Interpolation in the Guruswami-Sudan Algorithm. *IEEE Transactions on Information Theory*, 56:4341–4349, 2010.

[vzGG03] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ Press, 2003.

[WB86] L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes, December 1986. US Patent 4,633,470.

[Wil12] V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *ACM Symposium on Theory of Computing*, pages 887–898. ACM, 2012.

[Woz58] J. M. Wozencraft. List decoding. *Quarterly Progress Report*, 48:90–95, 1958.

[WS01] X.-W. Wu and P. H. Siegel. Efficient Root-Finding Algorithm With Application to List Decoding of Algebraic-Geometric Codes. *Information Theory, IEEE Transactions on*, 47(6):2579–2587, 2001.

[Wu08] Y. Wu. New List Decoding Algorithms for Reed-Solomon and BCH Codes. *IEEE Transactions on Information Theory*, 54(8):3611–3630, 2008.

[Wang^3] L. Wang, Q. Wang, and K. Wang. A Lattice-Based Linear Shift Register Synthesis for Multisequences of Varying Length. In *IEEE International Symposium on Information Theory*, pages 1751–1754. IEEE, 2008.

[YK92] K. Yang and P. V. Kumar. On the True Minimum Distance of Hermitian Codes. In *Coding theory and algebraic geometry*, pages 99–107. Springer, 1992.

[Zeh13] A. Zeh. *Algebraic Soft- and Hard-Decision Decoding of Generalized Reed–Solomon and Cyclic Codes*. PhD thesis, Universität Ulm, 2013.

[ZGA11] A. Zeh, C. Gentner, and D. Augot. An Interpolation Procedure for List Decoding Reed–Solomon Codes Based on Generalized Key Equations. *IEEE Transactions on Information Theory*, 57(9):5946–5959, 2011.

[ZW11] A. Zeh and A. Wachter. Fast multi-sequence shift-register synthesis with the Euclidean algorithm. *Advances in Mathematics of Communications*, 5(4):667–680, 2011.

[ZWB12] A. Zeh, A. Wachter, and M. Bossert. Unambiguous Decoding of Generalized Reed–Solomon Codes Beyond Half the Minimum Distance. In *International Zurich Seminar on Communications*, 2012.